The Pragmatic Programmers

TURING 图灵交互设计丛书

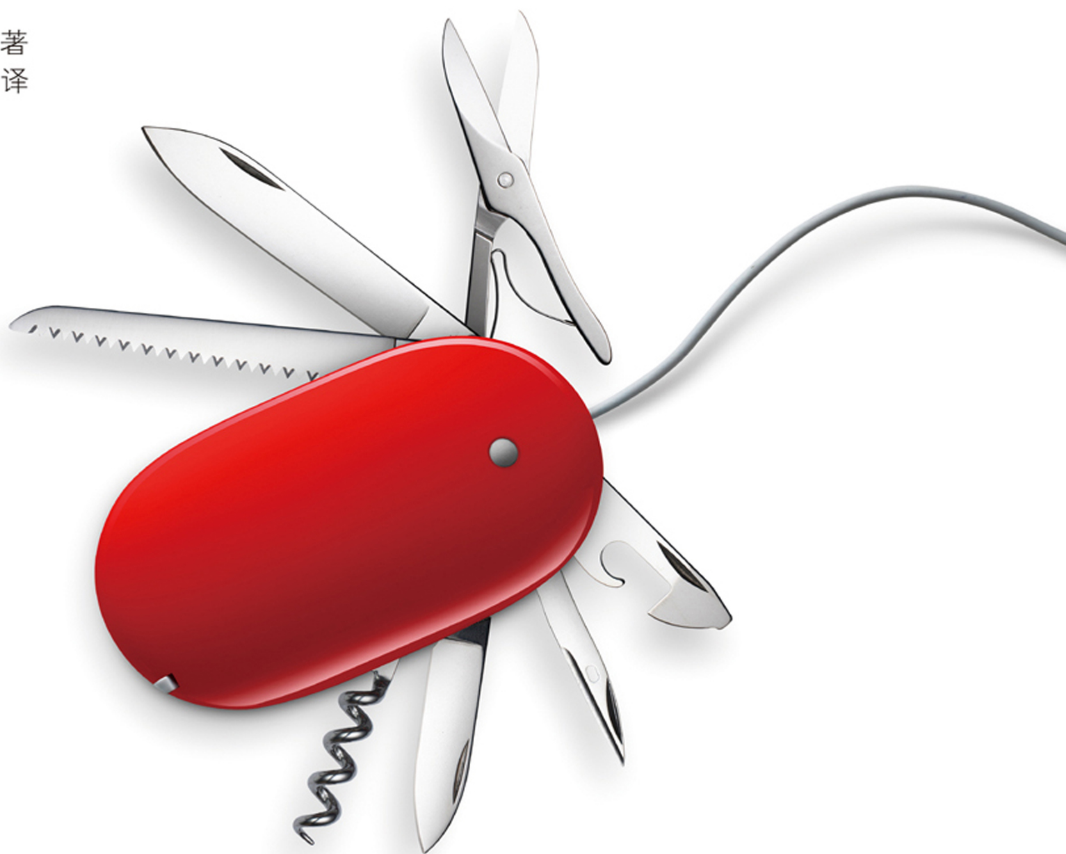
Designed for Use

Create Usable Interfaces for Applications and the Web

亲爱的界面

让用户乐于使用、爱不释手

【瑞士】Lukas Mathis 著
王军锋 杨 蕾 曾小进 译





人民邮电出版社
POSTS & TELECOM PRESS

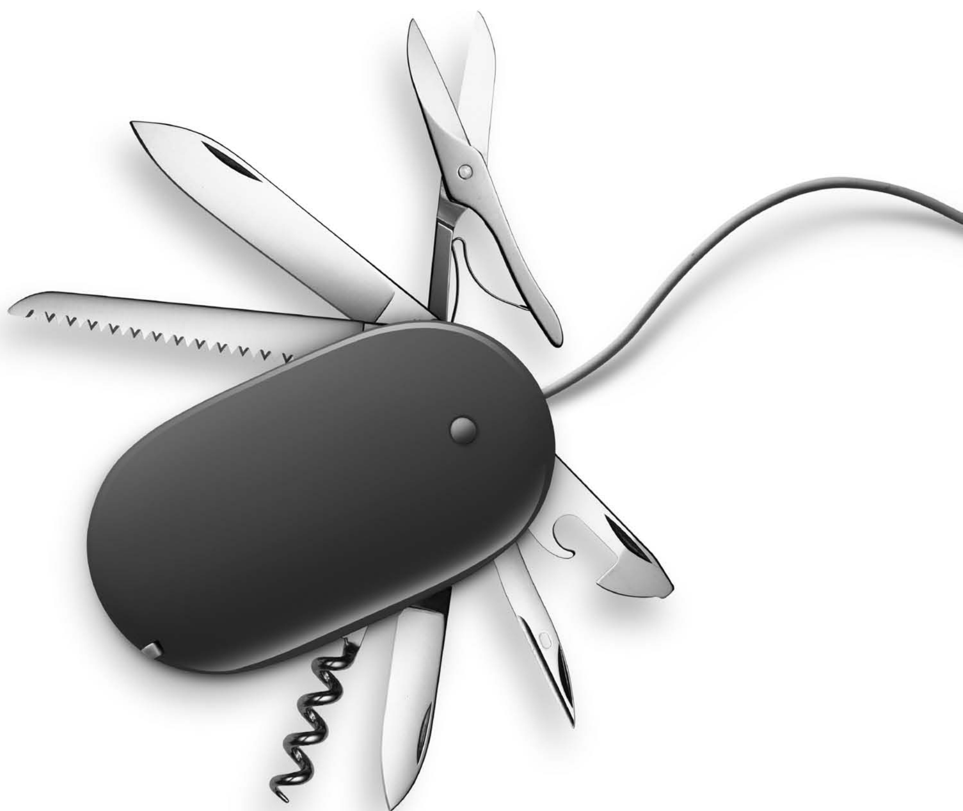
TURING 图灵交互设计丛书

Designed for Use

Create Usable Interfaces for Applications and the Web

亲爱的界面

让用户乐于使用、爱不释手



【瑞士】Lukas Mathis 著
王军锋 杨 蕾 曾小进 译

人 民 邮 电 出 版 社
北 京

图书在版编目（C I P）数据

亲爱的界面 : 让用户乐于使用、爱不释手 / (瑞士)
马西斯 (Mathis,L.) 著 ; 王军锋, 杨蕾, 曾小进译. --
北京 : 人民邮电出版社, 2012.11
（图灵交互设计丛书）
书名原文: Designed for Use:Create Usable
Interfaces for Applications and the Web
ISBN 978-7-115-29639-9

Ⅰ. ①亲… Ⅱ. ①马… ②王… ③杨… ④曾… Ⅲ.
①程序界面－程序设计 Ⅳ. ①TP311.1

中国版本图书馆CIP数据核字(2012)第242342号

内 容 提 要

本书主要介绍如何设计出具有可用性的应用程序和网站，不但让用户乐于使用，而且令其爱不释手。可用性是设计大厦的基石，作者详细介绍了如何将可用性融入设计、测试及开发的各个流程，如何优化设计过程、把握设计重点、提高设计效率。另外，作者在本书中给出了大量宝贵建议，传授了用户界面的设计与评估技巧，提供了独具匠心的设计视角。

本书适用于设计人员、开发人员、产品经理、专业软件开发工程师和用户界面设计师。

图灵交互设计丛书

亲爱的界面：让用户乐于使用、爱不释手

◆ 著　[瑞士] Lukas Mathis
译　王军锋　杨　蕾　曾小进
责任编辑　傅志红
执行编辑　张　霞
◆ 人民邮电出版社出版发行　北京市崇文区夕照寺街14号
邮编　100061　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京　印刷
◆ 开本：800×1000　1/16
印张：14.75
字数：349千字　2012年 11 月第 1 版
印数：1－4 000册　2012年 11 月北京第 1 次印刷
著作权合同登记号　图字：01-2012-4258号
ISBN 978-7-115-29639-9

定价：49.00元

读者服务热线：(010)51095186转604　印装质量热线：(010)67129223
反盗版热线：(010)67171154

版权声明

对本书的赞誉

这本书对软件用户体验设计全过程进行了百科全书式的描述，有最新最好的实践案例、内容翔实的设计方法。相见恨晚啊！

——凯斯·朗，Skitch公司COO和交互设计师

写可用性话题很难摆脱过分学院派的缺点，但卢卡斯做到了。如果你熟悉基本的可用性概念，又想了解更多的内容，这本书是必读佳作！

——乔恩·贝尔，Windows Phone的交互设计师

卢卡斯深刻分析了可用性、用户体验、UI 设计等方面为人忽视的许多问题，这是一本必不可少的、权威且极具启发性的著作。

——保罗·尼夫，Neave交互设计公司交互设计师

本书读来如瑞士巧克力般令人舒畅而愉悦，又如樱花盛开般华丽而动人，是所有学习可用性知识的人的必读宝典！

——迈克尔·D. 特鲁默，Appway公司高级管理人员

好好利用这本书吧，它能极大地提升你的设计！

——大卫·奈夫，Visionaer创意总监、设计管理者

译　者　序

计算机技术和网络的发展实现了许多前人未敢奢望的事情，同时也为我们的生活带来了诸多便利，但如何使用那些采用了这些技术的产品（无论是软件产品还是硬件产品）却成了最大的问题。人们经常被纷繁复杂的产品功能、前后不一的产品行为、缺乏反馈的界面设计等搞得晕头转向，有时甚至找不到产品的某项功能，即使找到了又不知道如何使用，知道如何使用却又不知道任务究竟完成得如何，任务完成了却又不知道如何退出……此类问题比比皆是。

由此，相关领域的学者开始对产品可用性、易用性、有效性等方面进行研究，提出了很多设计原则和方法以提高产品使用效率，提升用户使用产品的体验，促进和谐人机交互。国外出版社也出版了大量关于软件产品可用性设计、用户界面设计方面的优秀著作，本书的原版 *Designed for Use: Create Usable Interfaces for Applications and the Web* 正是可用性设计领域内浩如烟海的著述中一颗耀眼的新星。

作者卢卡斯・马西斯在他的博客 ignorethecode.net 上发布了很多关于设计与可用性的文章，并为瑞士的游戏新闻网站 wisegamers.ch 撰写关于电子游戏的评论。很多网站都转载了他所撰写的文章，如 UXmagazine 和 Splashnology 等。他还创建了大量用于 UI 设计的在线工具，发布在 iphonemockup.lkmc.ch 网站上。他目前就职于瑞士的软件公司 Numcom，负责工作流管理软件的开发工作和用户界面设计工作。正是作者在用户界面设计和软件产品开发工作方面丰富的经历，才成就了这本关于产品可用性设计的权威著作。

书中的内容大致分为两类：包含有齿轮样标记、关于设计技术的章节和包含有灯泡样标记、关于设计创意的章节。作者在关于技术的章节内介绍了可用于不同设计阶段、行之有效的用户界面设计技术，诸如工作观察与情景访谈、用户模型、编制文档、卡片分类、草绘与原型、纸质原型测试、远程和现场可用性测试等。在关于设计创意的章节里，作者根据自已多年的设计经验和相关的设计理论，向读者提供了激发设计灵感的多种方法，并给出了相关的设计建议。作者把用户界面设计分为三个主要阶段，即研究、设计、实施，并把全书的内容根据这三个阶段进行排序，方便读者的查阅。你可以从第 1 章开始，循序渐进地品味作者关于用户界面设计的独到见解，也可以根据自己所处设计阶段的需求，选择阅读那些需要了解的内容。感谢作者为我们奉献了如此优秀的一本著作，也感谢作者对本书内容编排设计的良苦用心。

西南科技大学的王军锋翻译了本书第 1 ~ 26 章及序言等部分，并负责全书的统稿工作；杨蕾

和曾小进共同翻译了第 27 ~ 36 章。

感谢图灵公司的编辑李松峰,他所出版的译著让我无比敬仰,我在为图灵公司翻译的过程中,向他学习到了很多关于技术类书籍翻译的技巧。感谢本书的编辑,她们为本书的出版付出了不少心血。感谢所有为本书的翻译出版提供帮助的人们!

在翻译的过程中,译者已及时对原著中出现的些许印刷错误作了订正,在此不一一指出。

鉴于译者能力有限,译文难免会出现错误和纰漏,希望各位同行和专家予以批评指正。

写在前面

本书的读者群是视觉设计师和程序员，但内容并不是视觉设计，也与程序代码无关，而是关于使用你所创造的产品的用户，这比前两者更为重要。

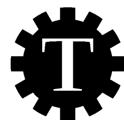
如果不能为人所用，再优秀的产品也毫无意义。你可以创造出最漂亮、最坚固、最优美的画笔，但如果没人用它来画画，那么你的努力就是徒劳的。

这本书能帮助你创作出人们想要的产品（应用程序或网站）。

本书的章节可以分为两类：技术章节和创意章节。每个技术章节都会介绍一种特定的技术，比如故事板（storyboarding）、可用性测试（usability test）和纸质原型（paper prototyping）等，将这些技术应用于设计过程，可以让你创作的产品更加人性化。技术章节主要介绍可以使用的具体技术，这些技术是设计师工具腰带中必不可少的工具。

而创意章节介绍的则是一般的创意和概念：如何撰写可用的文本，设计作品的视觉效果如何，何时使用动画等等。这些章节主要介绍设计过程中需要考虑的东西。

技术章节



技术章节的第一页都有齿轮样的标记。

所有技术章节的结构都一样。由于并非任何技术都可用于所有设计情况，因此，我们首先给出了技术适用的各种情境，然后对该技术的概念及使用方法进行了阐释。大部分技术章节的末尾，是该技术应用于阅读过程中所设计的虚拟应用程序时的具体示例。

鉴于 Twitter[①]的应用[②]（apps）在我们这一代学习创建应用程序上的先锋地位，我们将在技术章节设计一个 Twitter 的应用。为了增加趣味性，我们并没有设计一个普通的 Twitter 应用，而是面向那些需要更新自己公司 Twitter 账户的用户，设计了一款虚拟应用程序产品，我们称之为 BizTwit。

① 你不知道 Twitter 为何物（这很有可能，因为你可能是在 2053 年读这本书，届时，人们可能通过阅读神经系统来相互交流）？ Twitter（http://twitter.com）是一项非常流行的互联网服务，人们通过它来发布简短的文字信息——tweets，并订阅其他人的信息。

② 此处的“应用”指第三方应用程序，如未作特殊说明，以后相同情境均指此含义。——译者注

大家可把技术章节看作食谱，既可以按照顺序从前往后阅读，也可以深入钻研某个特定的主题。为此，这些章节都是篇幅简短，一语中的的，而且还提供了一些源自本书、其他书籍或是网络上的参考内容。

创意章节

技术章节主要介绍特定的技术及其使用方法，相比之下，创意章节就要宽泛一些了。这些章节介绍的概念，更多是作为灵感的来源，而不是死板严格的规则。某些章节会涉及某项技术或是技术章节的某些内容，但这些章节更侧重于一般概念，比如：设计要多切合实际、如何能更有效地使用动画、什么是交互模式、我们可以从视频游戏中学到些什么。

创意章节的第一页都有灯泡样的标记。



这些章节中所介绍的创意并不一定能应用于你正在从事的项目。因为从某种程度上说，人具有不可预测性。在使用你所创建的产品时，人们并不总是像你预期的那样进行操作，也不总是按照你所预测的规律行事。

为了说明人们的行为如何与预期大相径庭，让我们来看一个用户界面设计之外的例子。假设你关注公共卫生和安全，那么你会从哪里入手呢？鉴于每年有成千上万的骑自行车的人在交通事故中受伤，自行车安全或许是一个很好的切入点。

研究表明，头盔有助于防止骑自行车的人受伤。因此，让人们带上头盔可以降低受伤的概率，进而提升人们的健康和安全系数。预期结果似乎很明显：人们在自行车事故中受伤，而头盔可以保护人们不受伤害，所以骑自行车戴头盔的人不会受到伤害。那么问题解决了——强迫人们佩戴头盔。

书中网址的输入

本书内有很多网址，有些网址很长。如果阅读的是印刷书籍，那么把书中的网址键入到网页浏览器中可能会很麻烦。为了让这过程更简便一些，我建立了一个网页 http://designedforuse.net，里面包含了本书中所有较长的网址。有了它，读者就不用手动键入这些较长的网址，而只需要键入 http://designedforuse.net，然后点击里面的链接就可以了。

过去的数年间，人们已颁布了大量关于佩戴自行车头盔的法案。然而，这些法案并没有达到预期的效果。

2009 年，来自澳大利亚麦克里大学保险研究学院的Piet de Jong开展了一项名为“强制性自行

车头盔法案对健康的影响”[1]的研究，评估了此类法案的实施效果。他发现人们并不喜欢戴自行车头盔，如果强制骑自行车的人佩戴头盔，很多人就干脆不骑自行车了。

该调查结果让 de Jong 做出了以下推断：总的来说，自行车头盔法案实际上对社会健康产生了负面影响。不错，这些法案是可以防止一些人受伤，但对于那些完全放弃骑自行车的人而言，这些法案却对他们的健康产生了极大的负面影响，因为通常这些人会转向汽车。

归根结底，没有人在颁布法案之前邀请人们参与测试。受法案影响的人做出了法规制定者完全意想不到的反应。

在设计用户界面时，你将经常遇到类似的情况。设计方案的变更并不总能产生你所预期的效果，有时甚至会产生负面效果。

在阅读本书中介绍的创意和法则时，希望读者能谨记这一点。在设计用户界面时，你可能会竭尽全力设计出你认为非常有效的界面；你也可以遵循设计法则，做出那些具有显著可用性的设计方案。但用户依然会让你惊讶不已。他们以各种离奇的方式误解你设计的用户界面，在你的网站上摸不着东西南北，他们的行为完全不可预测，看上去毫无逻辑可言，他们无法完成你认为非常简单明了的任务。

不要以为在某款产品的设计中应用一系列可用性准则，就可以得到可用的设计结果。在设计用户界面时，的确要考虑常规思维，但不要过于依赖它。你需要了解设计法则，但如果能改进产品，不妨打破常规。关键在于，不要完全按照我告诉你的方法去做，而是要把书中的内容当作创意的源泉，而且要经常测试设计结果。

本书的组织结构

书中的章节基本上没有按照典型的设计流程进行排列，而是分成了三个部分，也就是产品开发的三个阶段：研究、设计和实施。

研究

尽可能早地进入设计阶段，开始设计产品（如果你是一个程序员，甚至会开始编写代码），这一点非常诱人。在某些情况下，这样做也无妨，但通常在设计之前最好还是开展一点研究工作。搞清楚产品的用户对象是谁，你所要解决的问题是什么。

设计

这时你要考虑如何解决用户的问题。设计出方案，并在编写代码之前对其进行测试。在纸上修正错误比通过代码要容易得多。

从设计的角度来看，这一阶段可能是整个开发过程中最重要的部分。因此，它也是本书

① 该研究参见 http://ssrn.com/abstract=1368064。

要着重介绍的内容。

实施

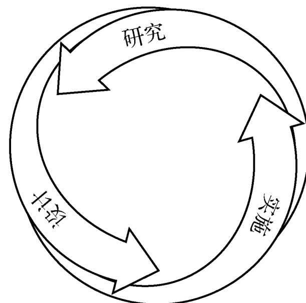
　创建产品，并持续进行测试。这一阶段要回答的问题包括：前期假设是否正确？设计方案是否有效？产品运行时人们如何与它进行交互？方案的实施结果是否满意？产品如何对错误和真实的数据进行处理？它的运行表现是否够好？

把创意章节放在何处这一问题比精确的科学研究还要困难。我把这些章节放在了读者最有可能看到它们有用的地方。但在很多时候，大部分创意都是适用的。这种组织方式比较适用于与技术相关的章节。

作者为每个技术章节引入了一个如下所示的时间标记：



这些时间标记可以帮助你了解哪项技术最为重要，哪项技术最常用。上图所示的标记表示该技术通常用于产品开发过程中“实施”阶段的前期。但很多技术适用于设计的不同阶段。书中的时间标记是为了说明技术的应用情景，而不是严格的法则。



这种标记方法使产品开发更像是一个典型的线性流程，从研究到设计，再到实施。但通常情况下，设计过程是迭代进行的。开发流程更像是一个循环过程，如右图所示。

但是，我们通常都认为产品的开发是一系列线性过程的迭代，因此线性的时间标记方法更易于理解。

最后一点

在阅读之前，还需要强调的一点是，网络上有本书的专门网页[①]，上面提供了论坛和勘误页面。当然，在本书成文之时，勘误页面仍然是空的，但当你读到这本书时，该页面可能已经有内容了。好了，闲话少说，现在就让我们言归正传！

① 参见 http://www.ituring.com.cn/book/839。

致　　谢

在撰写本书的过程中，很多人为我提供了帮助。在此，我将感谢所有提供帮助的人，但如果忘了感谢你，请接受我最诚挚的歉意，我将用一盒瑞士巧克力弥补自己的失误。

既然谈到了失误，本书中的所有错误，当然都是我个人造成的，与我要感谢的所有人都没有任何关系。

首先我要感谢，也最应该感谢的是我的编辑吉尔·斯滕伯格，是她把我语无伦次的表述变成了规范准确的文字。

感谢众多的技术编辑，他们纠正了本书的错误，并给出了大量非常好的反馈和建议。他们的名字如下（排名不分先后）。

凯斯·朗，skitch.com 的交互设计师。他的博客地址是 http://www.uiandus.com。

乔恩·贝尔，一个经验丰富的 Windows Phone 设计者。他的个人博客是 http://www.lot23.com；设计博客是 http://www. designdare.com。

麦克斯·斯滕贝亨，常驻企业的图形和 UI 设计师，他为豪华游艇上所用的软件设计了 UI 和图形元素。他的博客地址是 http://maxsteenbergen.com，Twitter 账号是@maxsteenbergen。

夏洛特·西蒙兹，诗人兼短篇故事撰写者。你能找到更多关于她的《世界上开得最快的花》一书的信息[①]。

邓肯·威尔科克斯，一个为 Mac、iPhone、iPad 编写软件的人。他的博客地址是 http://duncanwilcox.com。

克里斯·克拉克，交互设计师，一个怪才。他的网站地址是 http://releasecandidateone.com。

莎米拉·艾格尔，2012 年，她将在苏黎士大学完成心理学方面的研究。除了编辑本书内容之外，她还帮助我开展了一些研究工作。

大卫·奈夫，瑞士的一家企业形象设计公司的管理人员之一，创意总监。他撰写的文章可见 http://www.wisegamers.ch。

① 参见 http://www.victoria.ac.nz/vup/2008titleinformation/worldsfastestflower.aspx。

希比丽·阿瑞格，一个软件开发专家，她为银行编写风险管理工具。不幸的是，我还没有找到她的博客地址。

迈克尔·特鲁默，Numcom 软件公司 Appway 项目经理。你可以在 http://ch.linkedin.com/in/trummer 找到他，关于他对开发商业应用程序的见解，参见 http://twitter.com/DrummeratWork。

我还要感谢克里斯·普鲁特（在 http://replicaisland.blogspot.com 可以找到他），承蒙他的惠允，我在撰写第 26 章时使用了他制作的 Replica Island 游戏中玩家死亡位置的热成像图。同样感谢克莱顿·米勒（参见 http://rclayton.net），承蒙他的惠允，我在撰写第 19 章时使用了他提出的“延迟的被动确认”模型。

感谢 http://zuendung.ch 上的 Amadé Fries 允许我在第 23 章中使用了她拍摄的两张汽车照片。

感谢罗伯特·J.米克尔，他参与了山猫游戏机的设计，感谢他回答了我提出的关于山猫电子游戏机控制台的开发问题（参见第 1 章）。你可以在 http://www.mical.org 找到他。在撰写第 25 章时，Bohemian Coding（http://www.bohemiancoding.com）的皮耶特·奥姆威利与我分享了他从 DrawIt 中移除某些功能的经历。

感谢阿曼达·基弗，她帮助我开展了本书中的心理学研究。

很多人允许我在撰写本书的过程中引用了他们的书或者博客，不胜感激。

还有很多人允许我在本书的图片中使用了他们的 Twitter 图片、账号名称和所发布的微信。按照字母顺序排列，他们的 Twitter 账户名称为：@7WP、@aidanhornsby、@AlphabUX、@CoryCalifornia、@designdare、@fetjuel、@gauravmishr、@ienjoy、@jonbell、@larryv、@lorentey、@louije、@maxsteenbergen、@neave、@oliverw、@shawnblanc、@thibautsailly、@timeboxed 和 @workjon，感谢你们！

最后，感谢达玛莉丝·基弗，她拍摄了本书所用到的一些照片，并且她个人也出现在了其中的一些照片上。

目　　录

第二部分　设　计

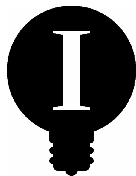
Part 1

第一部分

研　　究

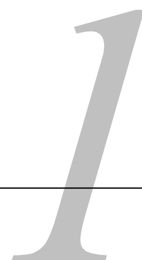
本书第一部分主要介绍与研究相关的内容。你将了解到为何研究如此重要、什么样的研究比较有效，以及研究过程中要避免哪些事情。除此之外，你还将学到如何观察人们在实际生活中的行为，如何对用户进行访谈，以及如何使用用户模型记录研究结果并找到产品设计的重点。最后，你将学会如何用卡片分类法构建产品。

那么，让我们先来了解一下为何研究如此重要吧。



第 1 章

用户研究



介绍自己的设计流程时，设计师通常都说它是“以人为中心”或是“以用户为中心”的。笼统地讲，这表示设计师经常要考虑所设计产品的潜在用户，尽力为这些人创造出最好的产品。

如何才能到这一点？

这个问题看似简单，实际上却不好回答。好的设计通常都是从用户研究着手的。

我们如何才能发现人们想要实现的目标？如何帮助用户实现这些目标？最简单的办法莫过于直接问他们。虽然这样做有时会得到一些有用的信息，但一定要小心地评估人们给出的答案。

亨利 · 福特曾说过这样的话：“如果我直接问人们想要什么，回答很可能是‘跑得更快的马’。”人们有什么理由不这么回答呢？绝大部分人都不是产品设计师，不会花费大量时间思考真正的问题是什么（比如，他们必须不断照顾自己的马匹），也不会去想如何用新产品解决这些问题，他们只是找些应急办法（比如修个马厩，或雇个人来照顾马匹），然后就心安理得了，对真正的问题视而不见。他们不会寻求新的方法来彻底解决问题，而只是追求让现有的东西略微好一点点。

于是乎，人们通常都不知道我们该如何解决他们的问题，甚至都不能讲明白到底有什么问题。最为糟糕的是，人们根本无法预知自己是否会或者如何去使用我们为他们设计的产品。

雅达利“山猫”（Atari Lynx）的设计就是个很好的例子。20 世纪 90 年代早期，日本电子游戏公司任天堂（Nintendo）依靠其产品“游戏小子”（Game Boy）统治着手持游戏市场。



几乎每个孩子都有一台 Game Boy 游戏机，而那些还没有的孩子则将此列入生日愿望单，急切地盼望着自己生日的到来。另一家与任天堂旗鼓相当的电子游戏公司雅达利发现了这一商机，准备开发一款产品投入市场。

在与焦点小组交谈之后，雅达利决定设计出比任天堂的灰色小玩意儿更强大的游戏机。雅达利的设备加上了彩色屏幕和更快速的处理器，起名为“山猫”（Lynx）。很明显，光从名字上就胜过了柔弱不堪、发育不良的“游戏小子”。雅达利给“山猫”配了大机盒，因为焦点小组的人说他们喜欢大一点的游戏机。

焦点小组

焦点小组（Focus Group）是一种用户研究方法。研究人员把某一品牌的产品、一项服务、一个新设计方案、一台设备或一个相似的产品展示给一群人，然后记录下来这些人的主观反应和意见，以此来预测普通公众对产品的反应。

结果雅达利的游戏机惨败，没一个人想要“山猫”。

我联系到“山猫”的设计者之一罗伯特·J.米克尔[①]，与他谈起以上历史，他说：

> 我从“山猫”中得到的最有价值的教训是：永远不要相信焦点小组。我们对“山猫”进行了大量的焦点小组测试，尤其在产品的尺寸和外形方面。我们拿出了大量的模型，问焦点小组的人：“你们喜欢哪一个产品？感觉哪个产品最好？”这里面有大尺寸的产品和小尺寸的产品，以及超大尺寸的产品和最为小巧的产品，但每一次他们都选择了大尺寸产品。所有人都说：“大的，我要大的！这样我才能感觉到物有所值。”OK，那我们就把游戏机做大。但当“山猫”问世的时候，所有人的反应却是：“这么大？！为什么这个东西这么大？”情况真不妙。最初的“山猫”机盒内部空空如也！我们本该按照自己的直觉来设计产品，但我们没有，而是按照焦点小组的说法去做设计。这一步走错了。

事实表明，人们既不知道自己真正想要什么样的手持游戏设备，也无法正确推测该如何使用游戏机。虽然口上那么说，但其实他们并不想要体型巨大、功能强大的游戏设备。孩子们喜欢把游戏机放入自己的书包里，随时都带着它。“山猫”的体积太大了，而且功能强大的处理器和彩色屏幕耗电量大，一组电池甚至用不到四个小时。也就是说，充满电的电池等不到放学就没电了。更为糟糕的是，其功能强大的硬件价格非常昂贵，导致很多父母都不太愿意为孩子买这么个东西。

人们以为自己想要一个体积很大、拥有强大处理器和彩色屏幕的游戏机，因为他们以为在这

① 更多关于罗伯特·J.米克尔的信息，参见 http://www.mical.org。

么大的设备上玩游戏是件非常酷的事情。而实际上，他们想要的却是一款价格便宜、体型小巧的游戏机，这样可以随身携带，并且用一组电池就能玩很长时间。

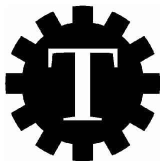


雅达利慌乱之中才又发布了“山猫”的小体积版本，但为时已晚。最后，雅达利仅仅卖出了 50 万台“山猫”游戏机，而任天堂却卖出了近 1 亿 2 千万台“游戏小子”。

现在，我们知道了目标用户是谁，但我们发现他们不知道自己想要什么，因此不能直接问他们，必须得自己去弄清楚。其实，目标非常简单明了。

发现问题	找到方法
找出人们目前正在做的事情	找到一种方法，使人们能更方便、更高效地完成正在做的事情
找出人们不喜欢做，但又不得不做的事情	找到一种方法，使人们不用再做自己不喜欢做的事情，或者至少使这件事情变得有趣一些
找出人们想要做的事情	找到一种方法，把人们想要做的事情变成现实

接下来的章节，将介绍挖掘用户需求的方法。



第 2 章

工作观察和情境访谈



工作观察和情境访谈分别是什么样的技术？

工作观察（job shadowing）和情境访谈（contextual interview）这两种方法用于观察人们的行为，发现他们需要帮助的地方，并查明产品该如何给用户提供帮助。具体来说，就是在人们工作时进行观察，以及就他们的工作与之交谈。

如果应用程序或网站的目标用户是特定的人群，而且你有能力与目标人群中的一部分进行交流，那么就可以采用这两项技术。比如，如果你正在为摄影师开发一款产品，那么请阅读本章；如果你创建的产品是为企业雇员服务的，并且你有机会与企业雇员进行交谈，那么请果断阅读本章。但如果对产品用户的定义尚不明确，或者你没有与用户进行交流的渠道，那么略过本章也无妨。

为什么这是个好主意？

用户和你是不同的。要试着去了解他们，了解他们的问题，这样你才会明白如何创造出他们真正可用的产品。

用户

天文学家兼作家克里夫 · 斯道尔曾有过经典一问：“为什么瘾君子和计算机爱好者都被称为用户？”的确，在这两种语境下都使用“用户”一词并不合适，但我也实在是找不到其他更好的词。因此我只能非常懊恼地认输，虽然极不情愿，但还得继续在本书中使用这个词。

不过，我会尽量尝试使用更恰当的术语。“人类”、“人”以及“客户”这三个词通常都可以非常恰当地替代“用户”一词。

是否存在前提条件？

前提就是，你需要明确产品的目标用户是谁，并且能够很容易地与目标人群进行交流。

2.1　观察目标人群

在上一章中，我们了解到大部分人都不是产品设计师。他们很难解释清楚究竟需要什么样的产品来实现自己的目标，并且通常无法准确地拿捏自己对产品的感觉。

因此，我们不能仅靠询问人们想要什么来解决问题，而应该自己寻找答案。工作观察和情境访谈正是用来完成这一任务的两种方法。

2.2　工作观察

由于人们根本不知道自己想要什么，因此比较好的办法就是直接观察他们的行为。“工作观察”是指在目标人群使用我们产品的场所内观察用户，其目的是找出产品应该如何帮助用户实现目标。这有点像可用性测试，但我们不是邀请用户过来测试产品，并告诉他们做什么，而是走访现场去观察他们的行为。

可用性测试的目标是找出与用户界面相关的问题。当你跟踪某个用户时，目标是找出需要创建什么样的产品，或是如何在更为基础的层面改进现有产品。

- 用户是否在某项任务上花费了大量时间？
- 用户是否在重复做某件事情？
- 用户是否在用权宜之计完成某些事情？
- 用户在做的某些事情是否很无聊，或者让他感到很厌烦？
- 用户是否被迫记住某项任务的操作步骤、技术事项或其他可以由电脑来完成的事情？
- 用户使用电脑时是否同时使用其他工具，例如在纸上列出清单或是使用计算器？

通常来说，你不能在用户工作时打断他，但如果你不知道他正在干什么，那就尽管问吧。

2.3　情境访谈

虽然说眼见为实，耳听为虚，但通过向用户询问恰当的问题，你仍然可以获得一些有用的信息。

在工作观察之后，花半个小时就用户所做的事情询问一些问题。你要做的是找出产品可以改进的地方。不要询问用户的观点，也不要把用户当作产品设计师，问一些太过专业的问题。

问题要紧密围绕用户所执行的任务，因此应该包括以下几个方面。

- 除了我今天看到的之外，你是否还经常要做别的什么任务？
- 你最常解决什么样的问题？
- 你为什么用这种方式做这件事（即采访者所看到的事）？
- 如果完成任务还需要一些别的信息，你会怎么做？
- 你经常打交道的人都有谁？你们是如何做的？
- 如果你需要的某个人没有上班，或者发生了其他问题，你需要做些什么？

需要注意的是，人们都是在没有经过深思熟虑的情况下回答以上问题的，因此你所得到的信息通常是不完整的，其中可能包含用户的个人感情色彩甚至是错误。尽管如此，情境访谈还是能为我们提供用户行为的概况，帮助我们找到还能从哪些方面改进产品。另外，通过访谈多个对象，我们还可以发现很多遗漏内容或是错误信息。

2.4 远程观察

如果无法访问用户，你可以请他们打开屏幕记录软件，记录下来大约半天的工作情况，然后将视频发给你。一开始，这样做可能会产生非常大的影像文件，但以下方法可以解决这个问题。

- 以较低的帧率来记录影像。绝大多数情况下，每秒一两帧图像的帧率就足以告诉你用户正在做什么。
- 不需要观察所有的细节，因此可以把影像分辨率缩小到屏幕分辨率的 30%，而且可以为其设定更高的压缩比。
- 可能根本不需要任何声音。
- 在用户工作的大部分时间，只有屏幕小部分内容会发生变化，因此最终的影像压缩效果会比较好。但一定要告诉用户关闭屏幕保护程序，特别是那些界面很复杂的屏保程序。

利用这些方法可以把 4 小时的影像压缩到 100MB 以下，但仍然能得到非常不错的视觉效果。

如果屏幕记录也无法实施，你可以给用户一个摄像机，让他装在工作场所内，记录其工作过程，记录完成之后把摄像机还给你就可以了。

快速浏览所记录的影像可以清楚地了解用户所做的事情，然后就可以询问他们具体的问题了。

2.5 情境访谈的局限性

人类可以做到其他动物无法做到的事情：他们可以想象自己处于某种假想状态下。哈佛大学的心理学家丹・吉尔伯特在他的著作《撞上幸福》（*Stumbling on Happiness*）中说道：“人类大脑最大的成就，就是可以假想出现实世界中根本不存在的物体或情景。正是具有这一能力，我们才得以展望未来。”

能够展望未来确实是一项非常棒的能力，但不幸的是，我们实在不擅此道，前一章中雅达利“山猫”游戏机的故事已证实了这一点。吉尔伯特还表明：“在假想某些事物时，我们总会犯一系列的错误。”

他对该现象的解释引人入胜，[①]但在本书中我不再展开，你只需要知道人类会在假想过程中犯错就可以了。

因此，你不能依赖人们的观点。当你对用户进行情境访谈时，要努力找出人们实际做些什么、需要完成哪些任务、遇到了哪些问题，不要期望他们能够告诉你问题的解决方案应该是什么样的，找出解决方案是你自己的任务。

人类不擅长预测自己使用产品的方式，这一点太糟糕了。更为糟糕的是，我们设计师也是如此，自己也不知道什么样的东西会有效，无法确定用户将如何使用新产品。[②]

用户研究能帮助我们做出更好的预测，但它并不能排除所有的不确定性。不要陷入无休止的研究之中。某些时候，你必须超越执念，做些尝试，然后看看用户是否认为它有用。

BizTwit 案例

对于 BizTwit 应用示例，我们花了半天时间与 ACME 公司的很多人待在一起，对他们进行了跟踪和访谈。该公司拥有一个 Twitter 账号，用来发布与企业相关的信息（如涉及公司产品的文章链接、与公司相关新闻的链接，还有客户关于公司言论的链接），并通过它与客户进行交流。

在 ACME 公司，我们分别面见了三个使用公司 Twitter 账号的人，试图找出他们使用 Twitter 的时间和方式。每次会面接近半天时间的时候，我们再和他们聊 30 分钟，问一些具体问题。

在访问的过程中，我们发现公司 CEO 上班的第一件事情就是浏览与本行业相关的文章。有时候遇到喜欢的文章，他就会把链接发布到企业的 Twitter 账号上。在其他时间，他会偶尔查看一下 Twitter，但极少发布或者回复任何信息。他现在每周只通过桌面的 Twitter 应用写两三条信息，但在访谈的过程中，他说自己不在办公室的时候，有时也会通过手机发帖。

第二个用公司 Twitter 账号发帖的人是该公司的公关代表。他的工作职责是让行业杂志对公司进行报道。当某个杂志报道了 ACME 公司，他就把这一信息发布到 Twitter 上。他大约每周在 Twitter 上发布一条信息。

最后一个用公司 Twitter 账号发帖的是一名工程师。当企业有重大活动，CEO 和公关代表忙于和客户交谈时，就由她负责报道。在这个过程中，她几乎全是通过自己的手机发布信息，介绍

① 你真的应该读一下丹·吉尔伯特的书，我保证你会喜欢的。如果你仍然作出了明显错误的估计，自认为不会喜欢吉尔伯特的书，那么你至少应该观看一下丹·吉尔伯特的 TED 演讲，参见 http://www.ted.com/talks/dan_gilbert_asks_why_are_we_happy.html。

② 认知科学专家唐纳德·诺曼曾在一次演讲中谈到过这一点和设计研究中的其他问题，参见 http://vimeo.com/12022651。

公司发布的新声明。她平常很少用 Twitter 发帖，但一旦使用，她会在一两小时内发布五六条信息。

在分别跟踪这三位用户半天时间之后，我们对他们进行了采访，此时就有一些新的想法冒了出来。公关代表说：

> 我遇到的一个问题是，CEO 通过手机往 Twitter 发帖时经常会打错字。如果我能有办法修改他的发言，就最好不过了。

工程师说：

> 在公司有活动时，平常发帖的人都很忙，这时就要靠我把新消息发布到 Twitter 上。但我不是个熟练的写手，常常会担心自己写的东西太过口语化，或者无法准确表达出自己的意思。

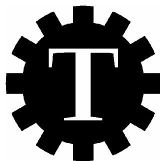
为了找出 Twitter 应用 BizTwit 应该解决什么样的问题，我们采访了很多用户，ACME 只是我们访谈的一家公司。然而，仅这三位受访者就为我们提供了非常有价值的信息，能帮助我们找出产品所要解决的问题。

提示

- 人们根本不知道自己想要什么，因此你必须走访他们，亲自观察他们的行为。
- 如果你询问人们具体的问题，你可能会得到有用的信息，但在下结论之前最好多访谈一些对象。

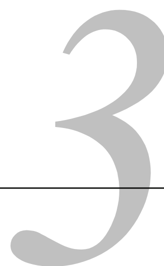
延伸阅读

肯尼德·鲍尔斯和詹姆斯·鲍克斯合著的《潜移默化：用户体验设计行动指南》（*Undercover User Experience Design*）一书中有一章专门谈到了此类用户研究方法。罗伯特·霍克曼在《一目了然：Web 和移动应用设计通识方法》（*Designing the Obvious*）一书阐述了工作观察和情境访谈的内容。



第 3 章

用户模型





什么是用户模型？

本章介绍创建和使用用户模型（Persona）的方法。用户模型是虚构的人，代表了特定的目标用户群体。

如果你是某个较大团队中的一员，正在进行用户研究，且整个团队需要就研究结果进行交流，那么用户模型会对你的工作大有帮助。但如果你是孤军作战，或者所在团队很小，那么略过本章内容也无妨。

为什么要用这项技术？

用户模型非常有用，因为讨论一个假想的人比讨论某个“细分市场”（market segment）简单得多。用户模型还能帮你找到产品设计的重点。

是否存在前提条件？

你必须在创建用户模型之前进行用户研究。

用户模型究竟是什么？

到目前为止，你可能已经完成了用户研究工作，找到了产品要解决的问题，知道哪些人群将从使用产品中受益。在设计产品时，你会经常用到这些信息。但该如何使用呢？对用户统计数据进行讨论是非常困难的，例如，哪一部分目标用户存在这个问题？这些用户的技能水平如何？

用户模型为你提供了一种方法，让你把从用户研究中获取的信息，合成为有限数量的假想人群。

艾伦·库帕首次在《交互设计之路：让高科技产品回归人性》（*The Inmates Are Running the Asylum*）一书中把用户模型作为一项软件设计技术提出。他这样描述道：

> 用户模型并不是真正的人。……它只是在整个设计过程中代表真实的人，是实际用户的假想原型。虽然用户模型都是假想出来的，但它的定义却非常严格精确。实际上，我们并不需要“虚构”用户模型，因为它是调查研究过程的副产品。

用户模型有助于设计交流。此外，它还有一些其他方面的优点。

- 它能够迫使你聚焦产品所要解决的问题。通过创建少量用户模型，你就可以清晰地定义出产品的用户。这样就剔除了那些不切实际的想法，比如你必须让所有人都对产品满意。
- 它使得对用户的讨论更加容易，而且通过深入考虑目标用户，它能帮助你改善设计流程，使其更加“以人为中心”。

3.1 用户模型的缺陷

使用用户模型的目标是使设计过程更加“以人为中心”，但要意识到它也存在很多问题。

用户模型的弹性太大。用户模型实际上是假想的人，无法为自己进行辩护。因此，它有时会强化预先定下的结论。如果把假想的人作为目标用户，你总是能够想象出一个场景来说明你当前所持的观点是合理的。

用户模型给人的印象是“以人为中心”，实现这一点不需要任何人与现实中的用户有所接触。它可以成为丝毫不“以人为中心”的设计流程的遮羞布。用户模型能让设计师免于所有艰辛的工作，比如不愿意走出去与实际用户接触、不去测试设计决策的有效性等。

创建用户模型是一件非常耗时的事情。从所有的用户研究资料中提炼出能代表目标用户的特定人群会花费你大量时间。你还要回顾用户研究的背景故事，然后讲述给所有参与设计过程的同事。有时，用户模型这一技术所能带来的益处无法弥补为此而浪费的时间。

讨论假想的人可能会让人感到不自在。假装故事中虚拟的“艾玛”是一个实际的人，一个想要使用你的产品的人，这件事不是设计团队内所有的人都愿意做的。

特别是对于比较小的设计团队来说，用户模型可能无法带来太多的益处。很可能团队中的所有人都已对目标用户是谁了如指掌。这时，没有太大的必要去创建一个虚拟的角色来帮助团队进行交流，而且用户需求可能已经极大地约束了产品设计，因为比较小的团队可能无法创建出能让较大用户群体满意的产品，即便他们非常想做到这一点。

即便如此，如果你能把以上事项铭记在心，用户模型还是一个非常有价值的工具。

3.2 创建用户模型

首先从情境访谈开始，与用户交谈。或许你认为目标人群人数众多，需要创建多个不同的用户模型才能涵盖这些人的特征。但随着访谈对象的增多，你会发现很多人都拥有相似的目标。因此，可以创建简化的用户模型，覆盖大部分目标用户群体的使用目标。创建的用户模型越少越好，

三个足矣，但根据所设计产品的不同，你可能需要创建更多的用户模型。

每一个用户模型都必须有明确的目标。为什么这类人会使用你的产品？他们想要通过产品实现什么样的目标？

然后，为用户模型加入一些相关设计细节。每个用户模型的技能水平如何？年龄如何？什么性别？他如何看待典型的一天？某个用户是否比其他用户更重要一些？如果是，是否要首先满足他的目标，即便这样做会对其他用户造成负面影响？每个用户都使用什么样的设备？比如，如果你设计一个网站，是否有人会通过手机浏览这个网站？

一旦确定了相关设计细节，就需要加入一些设计之外的、用户模型的个人信息。这样做的原因包括，首先，加入个人信息能让人更加容易记住用户模型，因为人类的大脑比较喜欢记忆私人信息。其次，个人信息能限制用户模型的弹性。我先前讨论用户模型缺点时曾说过，在用户模型中加入自己的观点很容易，因为用户模型无法为自己辩解。但你描绘的细节越多，就越难以加入个人的观点。最后，很可能你此时添加的某些信息在设计过程中会突然发挥作用。因此，你需要为用户模型加入诸如家庭、工作和个人兴趣爱好之类的信息。

最后，为用户模型设定一幅肖像，取个名字。

肖像图片一定要有特色，容易识别，但不能是某个人们所熟知的人的照片。所有来自图片库的图片或是简单的绘画图都可以。

为用户模型取个含有个人信息的名字是非常明智的（比如把用户模型所从事工作的首字母作为名字的首字母，或者把该用户模型的作用当作姓），但一定要避免名字带有负面含义，如哈里・骇客，也不能取个让人想起某个特定的真实人物的名字，如布兰尼・比伯。

3.3　使用用户模型

在讨论某个产品的设计时，我们往往会把目标群体统称为“用户”，然后考虑他们是否会喜欢产品能自动把照片上传到某个照片分享网站的功能，或是否知道如何使用照片上传功能等问题。

使用这种泛泛的概念很难弄清楚用户的需求，但有了用户模型之后就事半功倍了。

哪个用户模型想要把图片上传到照片分享网站？考虑到该用户模型的目标和所具备的技能，我们该如何设计产品的功能来实现他的目标？

不要再使用泛泛的概念了，我们应该谈论具体的用户模型。

3.4　用户模型无法代替用户研究

当使用用户模型这一技术时，我们容易假设“艾玛”知道如何使用自己的电脑，然后在为她

这样的人设计产品时，就不再考虑实际用户了，这样做是不对的。

用户模型有利于你和设计团队内的其他人员进行交流，对从用户研究中得出的数据进行评估，并使用该数据制定设计决策，但这些数据无法代替实际用户。你仍然需要让实际的人来测试你的设计，这样才能保证它行之有效。

BizTwit 案例

在上一章中，我们访问了若干在不同公司工作的人，目的是找出我们的 Twitter 商务应用能够解决哪些类型的问题。

现在，我们提炼出一些信息，建立若干个用户模型，每个模型代表一部分目标用户。图 3-1 给出了一个用户模型的示例。

这是一个非常精炼的用户模型描述，在实际的项目中，你或许还要另外加上一些细节。

马克·米勒（Mark Miller）	
工作	管理者，主要负责产品开发和客户开发，但有时也会关注市场
年龄	50岁
性别	男
技能	30年的使用经验（从Atari ST开始）使马克具有丰富的计算机和智能手机知识。他知道如何安装程序，但会把安装操作系统的工作留给企业的系统管理员
目标	希望能够通过自己的PC和手机向Twitter发布信息 他发布的信息通常包括一个指向某篇文章的链接和一些简短的评论。他希望能在其他人向公司的Twitter账号发布信息之前查看这些信息
生活背景	他有两个孩子，但都已经不在家里住了。他的妻子曼迪（Mandy）在一家文具公司工作，职务是销售主管。马克喜欢和她讨论与销售有关的问题，并且希望自己能深入到公司的销售事务中 他个人的行事风格比较低调。开一辆银色的奔驰车，经常穿一套深灰色西装。用联想的ThinkPad笔记本电脑，并且认为自己使用Android系统的手机而不是BlackBerry系统的手机是时尚之举 在闲暇时打壁球和网球

图 3-1　用户原型示例

提示

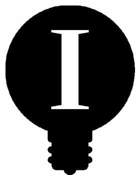
- 用户模型是假想的人，它代表某些特定的目标用户。
- 没有必要为每个人都建立一个用户模型。维护模型很浪费时间，而且很容易把你自己的想法加入到用户模型之中。
- 用户模型不能代替用户研究，它只能帮助你把用户研究的结果融入到产品设计中去。
- 用户模型能帮你评估从用户研究中得出的信息，把产品用户聚焦到某个明确定义的人群，并与设计小组就此进行交流。
- 使用用户模型有助于制定设计决策。不要再讨论宽泛的“用户”了，而是要讨论某个具体的用户模型，比如以下问题：产品的此项功能是面向哪些人的？用户模型想使用它吗？用户模型已经具备了哪些有用的知识？用户模型有些什么样的要求？产品的功能是否实现了用户模型的目标？

延伸阅读

艾伦 · 库帕在《交互设计之路：让高科技产品回归人性》和《交互设计精髓》（*About Face*）中都对用户模型有所阐述。

约翰 · 普鲁伊特的《用户模型生命周期》（*The Persona Lifecycle*）是一本专门介绍用户模型的好书。

拉里 · 康斯坦丁和露西 · 洛克伍德合著的《易用的软件》（*Software for Use*）也是一本介绍用户模型的好书。该书提出了一种不同的方法，作者称之为“用户角色模型”。



第 4 章

以行动为中心的设计

到目前为止，我们都认为“以人为中心的设计”总是最佳设计理念。但实际上，它并不见得总是最好的选择。还有一种方法是把人的行动作为设计的中心，对于很多产品来说，这种方法更有价值。

“以人为中心的设计”，即设计师要深入了解产品的潜在用户，为这些人量身设计产品。而“以行动为中心的设计”，则是为人的行动或目标量身设计产品。

“以行动为中心的设计”说来并不陌生，因为大部分设计都是以行动为中心的。比如，门把手就不是专门为某个特定用户群体设计的，它的设计是为了让开门的动作更加容易，更加一目了然。再比如汽车方向盘、电梯按钮、电脑上的大部分应用程序、你经常浏览的大部分网站等，这些产品的设计都是为了让某项或某一系列特定行为更加容易执行，让执行的方式更加一目了然。

“以行动为中心的设计”可能并不适用于所有的设计项目。有时，你可能要为有限的用户设计一款最佳产品。请对比图 4-1 中的两种电脑鼠标。



图 4-1　两种不同的电脑鼠标

图中左边的鼠标是与 Mac 电脑配套使用的。它的设计面向最广泛的用户群体，目标是让第一次触碰到这个鼠标的人都能像拥有二十年电脑使用经验的人一样熟练操作。为了实现这样的目标，它的默认配置只有一个按键，随便按下这个鼠标的任何地方都能触发这个按键。

这款鼠标非常简单，它不一定是最好的鼠标，但几乎所有人都能使用。

图中右边是 Cyborg R.A.T.[1]，一款专门为动作游戏用户设计的鼠标。它拥有 5 个不同的按键和一个可切换三种模式的开关，使用户可以为按键分配 15 种不同的命令。它有配套的可替换面板和掌托，以适应用户握住鼠标的不同方式，从而迅速调整鼠标的灵敏度。它还有可拆卸的配重物，用户可以自行加重或减轻鼠标的重量。它还是一款有线鼠标，因为无线连接会造成轻微的输入延迟，游戏玩家非常厌恶这一点。

所有这些功能使得这款鼠标成了目标用户的最佳选择，但也正是这些功能使其对那些非目标用户缺乏吸引力。一个拥有 15 种可编程动作、可拆卸配重的有线鼠标其实是不受大部分人欢迎的。

与前者相比，这款鼠标比较复杂。这意味着它是某些人的最佳选择，但也意味着很多人根本无法使用它。

应该以用户为中心，还是以行动为中心？这取决于产品的特性。不同的设计方法会产生不同的结果。尽早决定设计的焦点至关重要。

那么，该如何开展以行动为中心的设计呢？该方法最积极的倡导者之一唐纳德·诺曼认为，两者的主要区别在于态度不同。基于用户行为而不是用户个人进行思考时，视角就会有所不同。《易用的软件》一书的作者拉里·康斯坦丁和露西·洛克伍德认为："在最终的设计分析中，把用户理解为人远不如把用户理解为行动的执行者来得重要。"

以行动为中心的设计考虑的是用户行为，针对这些行为设计产品，而不考虑特定的用户或用户模型。它不是让产品去适应用户，而是以用户能够适应的方式设计产品。

提示

- 根据所设计产品的不同，把行动作为设计的焦点可能会更有意义。
- 进行用户研究，找出你所要支撑的用户行为，但不要为了特定人群而制造出使用行为。
- 严格评估用户反馈。有时，某些因素可能会让产品在某个特定用户群体中非常受欢迎，但也可能会让其他人不喜欢你的产品。
- 谨记，用户有能力适应你的产品，你不必总是使产品适应用户。

延伸阅读

唐纳德·诺曼在他颇受争议的文章《以人为中心的设计可能有害》[2]（*Human-Centered Design Considered Harmful*）中对"以行动为中心的设计"进行了阐述。

---

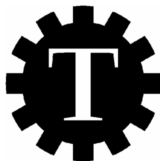
① 参见 http://cyborggaming.com。

② 参见 http://www.jnd.org/dn.mss/human-centered.html。

在《易用的软件》一书中，拉里·康斯坦丁和露西·洛克伍德提倡“以使用为中心的设计”，而不是“以行动为中心的设计”。

罗伯特·霍克曼在《一目了然》一书中对“以行动为中心的设计”有所描述。

当然，除了本章介绍的设计方法之外，还有一些其他的设计方法。例如，艾伦·库帕在《交互设计精髓》一书中介绍的“目标导向设计”。



第 5 章 文档编制



什么是文档编制？

本章将介绍使用手册（manual）、博文（blog post）、截屏视频（screencast）、新闻稿（press release）等内容。说得直白一些，这些东西能让人们对你的产品产生兴趣，能帮助他们学习如何使用你的产品。在开发之初就开始制作这些东西，这种做法被称为“逆向作业”（working backward）。

为什么要这样做？

尽早编制文档有助于对设计进行评估。如果你难以解释清楚某些东西，那么很可能是设计存在问题。

在设计过程中就开始编制使用手册，你就不太可能陷入术语、行话的囹圄，而是从用户的角度来解释产品。在某些东西上花费的时间越长，你就越难向那些与你有着不同知识背景的人解释清楚。

是否存在前提条件？

当然，你需要大概了解目标用户是谁，产品要解决的问题是什么。

5.1　使用手册

很多产品都附有介绍使用方法的手册。大部分使用手册都非常糟糕，这可不是件好事情，因为人们只是在不知道如何使用产品时才会打开这些东西。也就是说，用户在已经非常恼怒和不满时，还不得不阅读这些十分蹩脚、令人苦恼的手册。

当然，并非所有的使用手册都很糟糕，但只有高度重视才能制作出出色的使用手册，因此你要尽早开始思考并动手编写。

在设计过程中，你还不用考虑设计方案实施的细节问题。这有助于你从用户的角度看待使用

手册，因为用户也不知道关于产品工作原理的技术内容。

尽早开始编制使用手册还有另外一个好处，那就是它迫使你解释清楚产品的工作原理。没有什么能像这项工作那样迫使你思考设计的细节。如果某些东西很难解释清楚，那么它可能会很难用，你需要重新考虑它的设计。

那么，你该如何编写使用手册呢？

把使用手册当成是产品的一部分。出色的使用手册也是一项非常有用的功能，要像对待产品的其他功能一样对待使用手册。（如果编写代码和编写使用手册的工作同步进行，可以把对使用手册的检查纳入到产品的版本控制工作中去。）像设计产品的其他功能一样设计使用手册。试着回答以下问题：人们如何使用产品的使用手册？使用手册的结构应该是什么样的？它应该包括哪些内容？

产品使用手册不应该是“事后诸葛亮”。它是产品的一项重要功能，应该得到与产品的其他功能一样多的重视。

5.2 博文

使用手册非常重要，不过它不是向人们介绍产品的唯一途径。实际上，很多产品根本就没有使用手册。

博文是另外一种介绍产品功能的重要工具。与使用手册一样，它也能带来两方面的好处：帮助人们了解产品，帮助你发现产品潜在的问题。

在设计了产品的某项功能之后，你可以发布一篇博文告诉用户这项功能如何出色，该如何使用它来做些很酷的事情。你不一定要马上就公开发布它，但要试着写出一个草稿。你能毫不费力地解释清楚人们关注该功能的原因吗？你能描述出该功能有多简便吗？

对于类似的问题，如果有一个回答是 No，那么你就要用些招数来改变设计方案，让它更具说服力、更加有用、更易于说明。

顺便说一句，设计过程中的所有设计文档和实体模型你都应该保存好。这些东西或许可以帮助你写出一篇非常优秀的博文，题目为“该设计的诞生过程”。

5.3 截屏视频

截屏视频是个非常好的工具，可用来介绍新产品或现有产品的新功能。如果你正在设计一款产品，就应该考虑一下如何在截屏视频中对它进行展示。应该设计出什么样的卖点来吸引住用户？你是否能通过截屏视频说明某个具体问题的解决办法？你能否用线性叙事方法解释清楚产品的新功能？

如果已经有了产品原型，那么就制作一个截屏视频来解释其工作原理。当然也不一定要发布，仅仅制作截屏视频就能够让你意识到某些设计问题。（如果你想不到需要在截屏视频中解决什么问题，可能该功能根本就毫无存在价值。）另外，当产品完成时，为消费者公众制作截屏视频也是一项很好的举措。

5.4 新闻稿

你还可以在设计过程中为产品撰写新闻稿。你能在一条新闻中解释清楚产品，并让它听起来令人激动吗？

如果发布的新闻听上去毫无用处，也无法引起人们的兴趣，那么你的设计可能存在一些问题。

亚马逊就是从发布新闻开始设计产品的公司。亚马逊的 CTO 沃纳・维格尔说道："在发布产品之前就发布新闻，这样能弄清楚公众对产品的态度，而不是我们自己内部的观点。"[①]

你也可以尝试为自己的产品撰写一条广告语或是一个宣传口号。你能用一句话解释清楚吗？如果不能，为什么？

5.5 讨论产品的任务

无论你是撰写使用手册、博文，还是录制截屏视频，最好介绍一下产品能完成的任务，而不仅仅是它的功能。

文档通常只能简单地介绍产品的各项功能是如何运作的。除非你正在创建的是最基本的产品，比如一个闹钟，否则仅解释各项功能的运行方式是远远不够的。人们并不想学习如何使用产品的功能，他们想要学的是如何利用产品达成目标。用户都有自己的目标，他们之所以使用你的产品，是因为他们认为产品能帮助他们实现这些目标。

与其在博文中解释某个照片编辑器中图层的工作原理，不如说明如何为照片加上相框和很酷的效果，顺便把图层的使用知识贯穿其中。与其在使用手册里介绍文件处理软件中标签的用法，不如说明如何设计一封漂亮的信函，同时把标签的使用方法贯穿其中。

37signals 团队在他们的著作《把握现实》（*Getting Real*）中提到："如果你找不到令自己满意的语句来描述一项新功能或一个新概念，那么就为它编写一个简短的故事吧。"一定要避免涉及技术性的细节，仅仅把它说清楚就可以了。

此时要把自己当成一名老师，而不是一个编写技术材料的人。

---

① 关于更多此人的观点，详见 http://www.allthingsdistributed.com/2006/11/working_backwards.html。

BizTwit 案例

为了明确定义 BizTwit 的功用，设计团队已经撰写好了一篇博文，准备 BizTwit 设计一完成，就用它来介绍该产品。

> 现在已经有很多 Twitter 应用了，数不胜数。我们曾尝试了解所有这些应用，但下载这些应用浪费完了所有磁盘空间，所以最终还是放弃了。很明显，这个世界现在需要更多的 Twitter 应用！那么现在让我们来介绍自己的产品——BizTwit！
>
> 为什么人们还需要另外一款 Twitter 应用呢？
>
> 我们看到的大部分 Twitter 应用都是针对日常 Twitter 用户的。这些应用能让你关注其他人的信息，对他们发出的推特做出回应，并写下你自己的推特。更高级的 Twitter 应用还支持多账户登录和其他精彩的功能，比如与在线书签 Instapaper 进行整合应用。但所有这些应用都是针对常规用户的。
>
> BizTwit 不同，它的目标用户是那些要更新企业 Twitter 账户的人。它的设计目标包括：
>
> - 若干不同的人使用同一 Twitter 账户；
> - 每个人都能阅读并修改同一份草稿；
> - 不同的人拥有不同的权限。
>
> 在未来的日子里，我们将介绍这些功能，甚至更多。现在，请下载一份这个应用！

即便是如此简短的一篇博文，也能对你想要创建的产品做出一个很好的定义。

提示

- 编写产品使用手册、博文，制作截屏视频都能迫使你向其他人解释你的设计。这能帮助你发现设计存在的问题：如果某些东西很难解释清楚，那么它可能就不好用。
- 人们只有在遇到困难的时候才会阅读使用手册，因此使用手册一定要解决问题，而不是让用户更加郁闷。
- 发表新闻和博文能帮助你思考产品的设计目标，把重点放在所要解决的问题上。你能用一句话说清楚你的产品吗？如果不能，那么你要实现的功能可能太多了，或者根本没有解决什么具体的问题。

延伸阅读

布鲁斯·唐戈纳兹尼在他的文章[①]中提出了一些关于如何提升使用手册质量的想法。

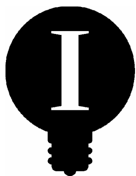
① 文章详见 http://www.asktog.com/columns/017ManualWriting.html。

如果你正在编写用户使用手册，并且需要一些方法对其进行测试，你可以在梅诺・德容和彼得・范德波特合写的文章《技术文档的可用性测试过程》（*Towards a usability test procedure for technical documents*）中找到答案，在 Google 图书或者《高品质的技术文档写作》（*Quality of Technical Documentation*）一书中可以找到这篇文章。

如果你想了解更多撰写文档的方法，请阅读下一章（第 6 章）。

苹果公司建议从“应用程序的定义描述”（application definition statement）开始设计产品，这一观点类似于“逆向作业”[①]。

① 更多内容参见 http://developer.apple.com/library/safari/documentation/UserExperience/Conceptual/MobileHIG/AppDesign/AppDesign.html。



第 6 章

文字的可用性



当我们谈及产品设计及其可用性时，往往首先关注视觉元素而会忽略文字。这可真是遗憾，因为文字是用户与应用程序或网站交互的一个主要渠道。实际上，在一些跟踪用户浏览屏幕时的视觉路径的测试[①]中发现，文字通常是用户寻找的第一个目标。

可用性专家雅各布·尼尔森在一篇关于如何为 Web 撰写文字的文章[②]中谈道：

> 在对网页进行意见反馈时，用户更多的是针对网页显示内容的质量和相关度进行评论，而不是针对导航或页面之类我们称之为“用户界面”的东西（相对于单纯的信息而言）。同样，当弹出一个页面时，用户会把注意力放在窗口的中心位置。他们首先阅读的是文字内容，之后才会看一眼标题栏或导航。

37signals 团队对此也持相同观点。在《把握现实》一书中，他们写道：“有好文字，才有好设计。”因此，应当把文字当作界面设计的一部分。

6.1　文字的重要性

你有没有尝试过网购电脑？有没有见过这样的网站，在说明电脑性能时，用的不是平白的语言，而是一大堆故弄玄虚的营销潮词？嗯……我该买一台什么样的电脑呢？带“MagSafe”接口的？有“一键影音”功能的？或者有“ThinkVantage”键的？还是有“面向企业设计的多功能一体可扩充 UltraTouch 面板”的？好吧，最后一个是我编的，但其他几个确有其事。[③]

文字是用户与产品交流的渠道。设计即是沟通，如果别人无法理解你所要表达的意思，那么他们就无法使用你的产品。

产品设计的各个方面都要使用文字：网站、使用手册上必不可少；与用户进行私下交流时，你也要用到；即便是编写代码中类的名称和备注，也会用到文字。因此，最好还是早点确定文字内容。

但是，你怎么知道该使用哪些文字呢？

---

① 关于视觉路径跟踪测试的所得成果，参见 http://joidesign.com/BlogRetrieve.aspx?BlogID=45&PostID=4981。

② 参见 http://www.useit.com/papers/webwriting/writing.html。

③ 那文·摩根（Neven Mrgan）还列出了另外一个网站，该网站使用了企业知道但用户根本无法理解的词汇，参见 http://mrgan.tumblr.com/post/3241126895/what-does-the-user-see。

6.2　人们根本不想阅读文字

刚刚介绍了文字的重要性，我们又提出这样一种观点，听上去可能有点奇怪，但无论你信不信，事实确是如此，大多数人都在尽可能地避免阅读文字。

你阅读本书，就已经证明了一点：你和大部分用户有很大不同。约翰·M.卡罗尔和玛丽·本斯·洛松在 1987 年的一篇论文“活跃用户的悖论”[①]中指出：“各个层次的学习者都试图逃避阅读。”在过去的 20 年内，这一点几乎没有任何改变。实际情况更为糟糕，美国艺术基金 2007 年开展的研究[②]表明，与以往相比，美国人不仅阅读数量减少，而且阅读质量也在下降。

或许你曾接到“我的打印机不工作了”之类的电话：

你的朋友：我急需打印这个文档，但我的打印机不工作了。

你：哦，究竟发生了什么事?

你的朋友：我试着打印文件，但打印机不工作了。

你：你有没有看到什么错误信息?

你的朋友：噢！是的，有错误信息弹出。

你：是什么?

你的朋友：我怎么知道？我把它点跑了，当然这样做根本没用。

北卡罗来纳州立大学心理学院有一篇名为 Failure to Recognize Fake Internet Popup Warning Messages[③]的论文，该论文分析了人们如何处理弹出的虚警信息。在结论部分，作者描述道：

> 从问卷调查得出的数据来看，遇到弹出信息时点击了 OK 按钮的人中，12%的人说是屏幕上显示的文字告诉他们应该这样做；23%的人说当遇到这种情况时，他们总是点击 OK；几乎有接近一半的人（42%）说他们这样做只是想让弹出的信息消失。

在点击按钮清除弹出的窗口时，人们甚至连看都不看一眼。这些东西非常讨厌，会打扰到人们正在进行的工作，把它们清除掉好像就是解决问题的办法。

遭受如此待遇的不只是弹出窗口中的文字。只要是人们认为自己可以略过的任何文字信息，他们都会这样做！

6.3　少用一些文字

既然用户根本不想阅读，那么最好还是尽量不要让文字打扰到他们。

① 论文参见 http://dl.dropbox.com/u/16760174/Papers/Paradox.pdf。
② 更多信息参见 http//:www.nea.gov/news/news07/TRNR.html。
③ 论文参见 http://media.haymarketmedia.com/Documents/1/SharekWogalterFakeWarning_publicationFinal_805.pdf。

例如，在用户将要做某些破坏性事件时不要发出警告，而是允许他们撤销自己的操作。

同样，如果发生了一项错误，你有办法让产品在不打扰用户的情况下自行恢复，那么就不要让用户知道。如果用户输入了一个不完整的网址，但你的网站已经得到了足够多的信息，知道用户将要访问哪个页面，那么就直接跳转到那个页面。如果你的应用程序尝试连接到某个服务器，但连接超时，在告诉用户有错误发生之前，让程序再次进行连接。只有当你的产品实在无法搞定某个问题时才告知用户。

如果人们无法理解某些用户界面，不要在界面上附加说明性的文字，因为这样做只能让界面更加凌乱，更加糟糕。

如果无可避免地要告知用户，那么就先假设用户根本不会阅读你的文字，再去设计用户界面。比如在按钮标签上使用动词，保证每个按钮都有特定的、排他的标签。不要把按钮的标签设为“Yes”和“No”，而应该设定为“删除文件”和“取消”。这样，用户不用阅读对话框中的文字，就知道每个按钮的作用。

6.4　使用可粗略浏览的文字

通常情况下，使用文字是无法避免的。想知道如何撰写文字，首先要知道人们是如何阅读的。

雅各布・尼尔森的研究表明，人们通常不会在网页上逐字逐句地阅读，而是粗略地浏览页面，寻找那些包含有用信息的文字片段。为了帮助人们实现这一目的，尼尔森给出了以下设计准则。[①]

- ❑ 使用对用户有用的词汇；
- ❑ 每段只阐明一个主旨；
- ❑ 将各段主旨置于首句，以便用户迅速做出决定是否阅读；
- ❑ 使用能概括段落含义的标题；
- ❑ 加亮关键词；
- ❑ 使用项目符号列表；
- ❑ 使用简短、日常化的文字；
- ❑ 以结论性的语句开始段落，并对段落内容进行概述。

6.5　不要啰嗦

用户为什么会阅读你写出的文字？因为他们认为这些内容有助于实现自己想要的目标。尽量给出简短、明确、含义清晰的语句，使用短小精悍的段落，不要浪费读者的时间。始终把他们的目标铭记在心。

---

① 关于这一问题，尼尔森发表了很多文章，参见http://www.useit.com/papers/webwriting/。

正如帕特里夏·怀特在《高品质的技术文档写作》一书中所说："作者们通常自以为，应该为读者呈现超出其所需的更多内容。"

当你撰写文字时，请问问自己："这句话对用户有用吗？"如果答案是否定的，那么就把它去掉吧。

6.6　语句表述清晰

不要使用那些模棱两可的语句，也不要使用那些只有读完整句话读者才不至于误解的语句。[①]

不等读完整句话，读者就会试着理解句子的含义。某些句子会对读者产生误导。思考以下标题前几个单词的含义：[②]

"Burger King fries the holy grail..."

为什么汉堡王会炸（fries）圣杯？当你读完整个句子时，才恍然大悟：

"Burger King fries the holy grail for potato farmers"

噢！汉堡王的炸薯条（fries）是种土豆农民们的圣杯。句子的含义依然不是十分确定，但已经相当明显了。

这种句子让读者大费周章，因为如果一开始就理解错了，那么就不得不返回到句子开头重新读。为了避免让读者产生迷惑，你要问自己：某个句子是否清晰明了，即便是只读一半？小小的改动就能让这类句子的含义非常明确，比如：

"Burger King fries are the holy grail for potato farmers"

你能找出以下句子存在的表述问题吗？

"Lukas told his editor that he would write a project plan to finish the book by the end of the month."

我是将在这个月的月底写出项目计划来，还是计划在本月底写完整本书？

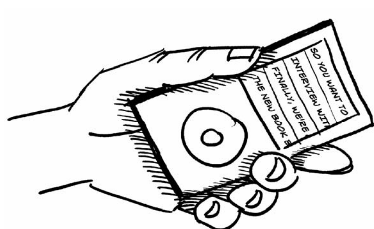
如果你写出的文字让读者读起来根本不费任何心思，那么用户才会愿意读这些文字，才能理解其中的含义。

清晰明确对标题而言尤为重要。使用和内容相关的词作为标题的开头，这样，即便是在小屏幕设备上，标题不能在一行完整显示，用户也能理解其含义。

---

① 这种句子被称为"花园小径"（garden path），来自于俚语"to be led up the garden path"（把某人引入歧途）。对这种句子的直觉性理解会让人产生疑惑，在读到句子的末尾时，读者必须返回来重新理解句子的含义。更多内容，参见维基百科（Wikipedia）http://en.wikipedia.org/wiki/Garden_path_sentence。

② 我在一个非常棒的"语言日志"博客上发现了这个例子，参见 http://languagelog.ldc.upenn.edu/nll/?p=1762。如果你对写作有兴趣，这个博客绝对值得一读。



糟糕的标题
在小屏幕设备上，
用户很难搞清楚被截
断显示的标题的含义。

6.7　不要以企业口吻书写

企业通常使用指定的书写风格，以加强所有文案风格的一致性。他们通常鼓励撰稿人使用第三人称和中立口吻，因此，不提倡在书写中体现个人风格。

不幸的是，在这些原则的指导下写出的文字空洞乏味，苍白无力。人们并不想阅读这样的东西，而且对那些必须撰写这类文字的人来说，写作过程也是让人心碎不已的。

如果只注重一致性，就会把所有人都变成最差的写手，还不如想办法来吸引读者。直接向读者表明来意，使用那些在谈话中听起来非常自然的语句。不要太正式，而是要与读者交谈。避免使用营销术语，必要时使用“你们”和“我们”之类的词（但在说“我们”时，一定要避免那种大人与小孩谈话时居高临下的口气）。

被动语态

人们在给出写作建议时，常常说不要使用被动语态。有时候确实如此，比如“点击备份（Backup）以创建一个拷贝”就比“拷贝可以通过点击备份而被创建”要好一些，但人们通常过分夸大了使用被动语态的弊端。比如，“不必四下寻找‘备份’按钮——你的文档会被自动备份”这样的语句并无大碍，虽然它使用了被动语态。实际上，把这种表述换成主动语态会平添出无用的词，使句子变得更差，比如，“程序将自动备份你的文档。”

我猜人们并不是说不要使用被动语态。毕竟，被动语态和主动语态的区别仅仅在于，句子的主语不是动作的发出者，而是动作的承受者。

人们真正的意思是，应该避免把句子的主语变得含混不清。人们常常用被动语态来为自己开脱责任，比如“有错误发生了”。避免使用含混不清的表达方式通常是个不错的建议。

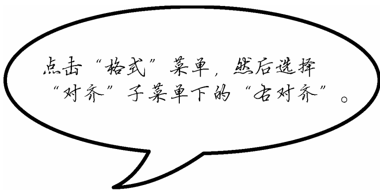
实端的协同传输通道能让用
户实现即插即用的电子协同效应。

请大声读出上页图片中的文字。你是否愿意阅读类似的东西？非营销部门的人能够发现其中的有效信息吗？

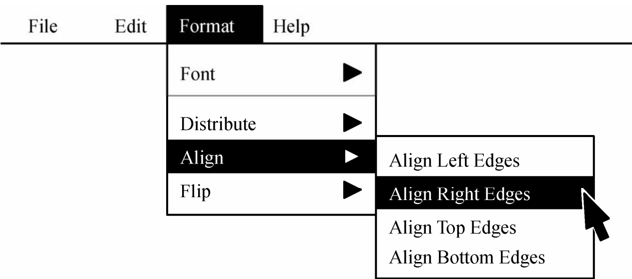
6.8　使用图片阐明要点

维护图像和屏幕截图的工作量很大。每个产品更新时，都必须检查所有图片，更新那些已经过时的。这容易诱使大家完全不使用图片，但是图片能让文字更具可读性，更易于理解。

使用图片有助于阐明要点。恰到好处的屏幕截图能省去好几段文字，也更易于理解。对于有些用户，使用插图甚至可以为简单的说明增色不少。



虽然以上表述非常清晰明了，但下面这幅图像却能让含义更加显而易见。



图像还能让文字更加引人入胜。面对着满屏的文字是十分痛苦的，但加上几幅图像之后，人们就会更乐于阅读。玛丽安·伯特兰等人的论文“心理学何用——一个消费信贷市场的实地实验”[①]表明，仅仅为某个信贷员附一张女性图片就能使贷款利息出现具有统计学意义的提高。不过，雅各布·尼尔森进一步发现，如果图片看上去像是来自于图像素材库，或只是作为填补空白之用，那么用户一样会对其视而不见，置之不理。[②]

6.9　使用人们能够理解的词汇

不要让自己的好恶影响到所撰写的内容。有人不喜欢使用“播客”一词，也有人不喜欢“博

① 参见 http://karlan.yale.edu/fieldexperiments/pdf/Bertrand%20et%20al_2006.pdf。

② 关于雅各布更多的研究成果，参见 http://www.useit.com/alertbox/photo-content.html。其他关于这一主题的研究，参见 http://uxmyths.com/post/705397950/myth-ornamental-graphics-improves-the-users-experience。

客”，还有人不喜欢用“精简版”一词描述免费版的 iPhone 应用。但其他所有人都会使用并可以理解这些词。人们知道什么是“播客”，什么是“博客”，也知道某个 iPhone 应用后面带有“精简版”一词表示什么意思。请使用人们能够理解的词汇，即便你不喜欢这样的词。正如《卫报》的前科学编辑蒂姆・拉德福特所说的那样：“没有人会因为你把某些东西变得更容易理解了而抱怨不已。”[①]

记住，阅读你的文字的人可能会使用完全不同的词汇。如果你了解用户，那么就专门为他们撰写内容吧。用户可能处于不同的年龄段，拥有不同的技能水平，也可能具有与常人不同的领域知识。在撰写内容时把这些因素考虑在内，才能让用户更易于阅读和理解。

最重要的是，一定要让所写的内容简单易懂。史蒂芬・金在《论写作》（*On Writing*）一书中说道：“在撰写内容时，你能做出的最糟糕的事情莫过于，由于羞于写下太短的句子而去卖弄词藻和追求长句。”他的经验是：“如果你想到的第一个词合适生动，那么就使用它。但如果你犹豫不决，反复思考，也能想到另外一个词。你当然能想到，因为总有很多同义词。但这个词很可能并不会如第一个词那样贴切，在表情达意上也可能难以匹敌。”

虽然金所说的是如何撰写小说，但是，何不让你写的内容也像史蒂芬・金的小说那样引人入胜，妙趣横生呢？

6.10 测试文字

文字是用户界面的一部分，因此你可以把文字测试列为常规可用性测试的一部分。但你能做的并不仅限于此，可用性专家安吉拉・考特尔建议还可用填空测验（cloze test）来对文本进行测试。[②]

在填空测验中，移除所撰写内容的部分单词，然后让参与者填上缺失的单词。图 6-1 是一个简单的示例。

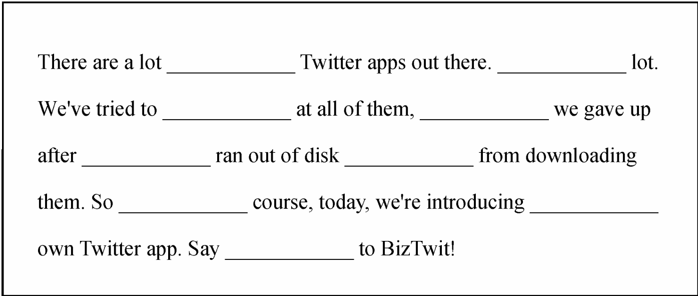


图 6-1 填空测试示例

① 关于蒂姆・拉德福特的《简单写作的宣言》，参见 http://www.guardian.co.uk/science/blog/2011/jan/19/manifesto-simple-scribe-commandments-journalists。

② 她关于测试内容的更多建议，参见 http://www.alistapart.com/articles/testing-content/。

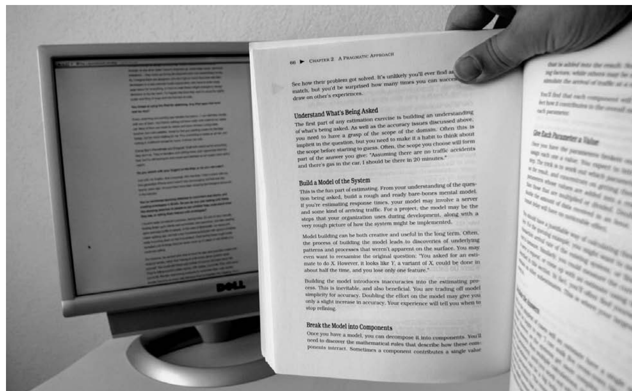
考特尔建议从产品界面中选出长度为 125 ~ 150 字的内容片段，然后每 5 个词留一个空。让测试者补上缺失的词。用填写正确的单词数除以缺失的单词总数，就是测试得分。

如果得分低于 0.4，那么就表明用户可能无法理解你的文字，就需要重新撰写；如果得分高于 0.4 但是低于 0.6，表明用户理解起来有点困难，你需要对文字进行修改；高于 0.6 就表示用户完全可以理解你的文字。

6.11　使用易读的文字

保证内容有用非常重要，但呈现内容的方式同样不容忽视。

使用大号字。虽然大部分人读书时都离书很近，但看电脑屏幕时距离要远一些。坐在电脑前审视你设计的网站或应用程序时，可以把书放在正常的阅读距离处，然后对比书中的文字和电脑上所显示文字的字体大小。[①]



如果发现电脑上的文字比书中的文字小很多，那么就把屏幕上的文字变大一些。需要注意的是，同一号字在不同的分辨率下显示的结果是不一样的。分辨率较低，文字较大；分辨率较高，文字较小。

如果你正在为手机创建内容，那么情况恰好相反。人们常常会把手机放在离眼睛很近的地方，所以你可以使用较小的字号。

① 我第一次看到这一方法是在奥利弗·雷彻斯坦（Oliver Reichenstein）的文章中。关于该文章，参见 http://informationarchitects.jp/100e2r/。



使用易读的字体。不同字体的易读性差别很大，甚至是同一字体族内的不同成员，相差也很大。选择一个好的字体非常重要。[①]

提示

- 如果可以，不要使用文字。
- 如果无法避免，使用精炼、含义明确且可以略读的文字。
- 使用短小精悍的段落，每段文字阐述一个主题。
- 保证文字具有吸引力和个性，而不是枯燥无味或过于专业。
- 用插图阐明要点，让文字更加平易近人。
- 使用大号字和易辨识的字体。

延伸阅读

雅各布·尼尔森写了很多关于为网页撰写内容的好文章。[②]如果你正在撰写产品使用手册，帕特里夏·怀特的《高品质的技术文档写作》一书中有很多有用的信息。乔尔·斯波斯基在《程序员的用户界面设计指南》（*User Interface Design for Programmers*）中谈到了有关阅读的内容。

自然科学作家卡尔·兹莫尔也写了一篇关于如何撰写优秀科学论文的文章。[③]蒂姆·拉德福特根据自己从事记者工作的经验提出了一系列撰写好文章的准则。[④]如果你对如何写出优秀的文章感兴趣，我强烈建议你读一下史蒂芬·金的《论写作》。

除此之外，你还可以读一下安吉拉·考特尔关于内容测试的文章。[⑤]

---

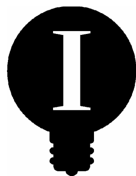
① 注意，研究表明，一款字体是否有衬线也会影响其易读性。如果你对与阅读相关的研究有兴趣，更多相关内容参见 http://www.alexpoole.info/academic/literaturereview.html。

② 参见 http://www.useit.com/papers/webwriting/。

③ 参见 http://blogs.discovermagazine.com/loom/2011/01/12/death-to-obfuscation/。

④ 关于这些准则，参见 http://www.guardian.co.uk/science/blog/2011/jan/19/manifesto-simple-scribe-commandments-journalists。

⑤ 参见 http://www.alistapart.com/articles/testing-content/。



第 7 章

用户界面设计中的层级结构

在思考网站的组织形式时，你会发现它们通常都使用了层级结构。人们有时甚至会把层级结构当作显性的用户界面元素使用，Google Directory 之类的网站上的“面包屑”导航元素就是个典型的例子。

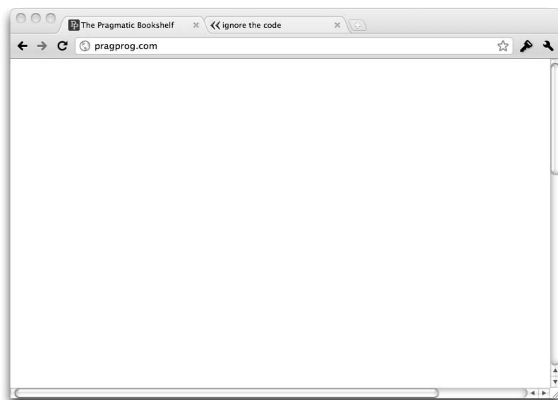
Software
Science > Biology > Bioinformatics > Software

这种导航元素可以告知用户当前页面在层级结构中所处的位置，并可以让用户“跳出”层级结构。

很多新闻网站的标题栏都显示出种类层级结构，但这些标题栏通常使用两、三层标签显示内容，而没有用上面的“面包屑”式布局。

U.S.	**World**	Europe	Technology	Business	Science	Opinion
Africa | Asia | Middle East | Japan						

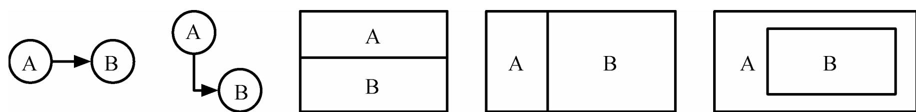
以上两个示例展现的是如何把整个网站或产品总体组织成层级结构，不过，其实每个单独的页面也都使用层级结构来组织内容。让我们看一下 Google Chrome 浏览器的截屏：



即便是这种非常简洁的用户界面，也包含有层级结构。例如，某个标签页下的按钮仅对同一页面下的内容有效。如果你点击“返回”（back）按钮，不会影响当前活动标签页之外的任何东西。但如果你点击的是窗口的“关闭”（close）按钮，那么整个浏览器窗口就会关掉，包括已打开的两个标签页。按钮仅对处于相同或下一层级结构的内容有效。

创建可视化层级结构

通常浏览一下任何用户界面，很快就能看出用户界面元素的层级结构。大多数西方用户的直觉是，层级结构应该是从左向右、从上向下、由外而内的。在以下几个示例中，A 是 B 的上一层级。



7

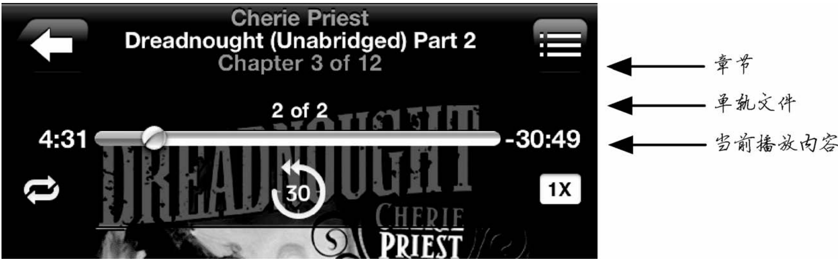
以播放有声图书中某一音轨文件的 MP3 播放器的界面为例，你应该如何安排以下元素：音轨文件编号、音轨文件内的章节编号（对于有声图书或播客来说，或许还包括章节的作者）和当前播放内容在章节中的位置。很明显，这些元素的相互关系如下所示：

音轨文件→音轨文件内的章节→当前播放内容在章节中的位置

下图是 iPod 的应用在 iPhone 上的显示结果。



章节编号和音轨文件编号的显示顺序反了，图中当前播放的音轨文件的章节编号显示在了音轨文件编号的上方，如下图所示。

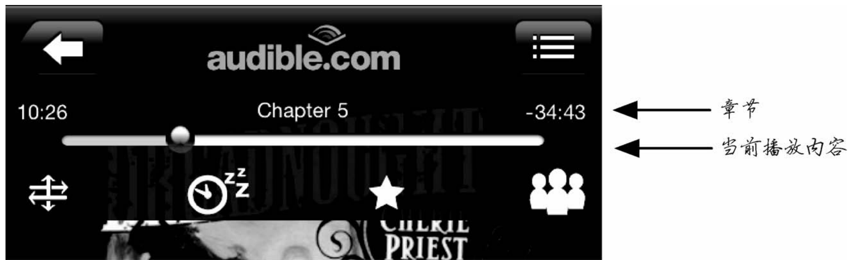


这种显示布局隐含着如下错误的层级结构：

当前音轨文件的章节→音轨文件→当前播放内容在章节中的位置

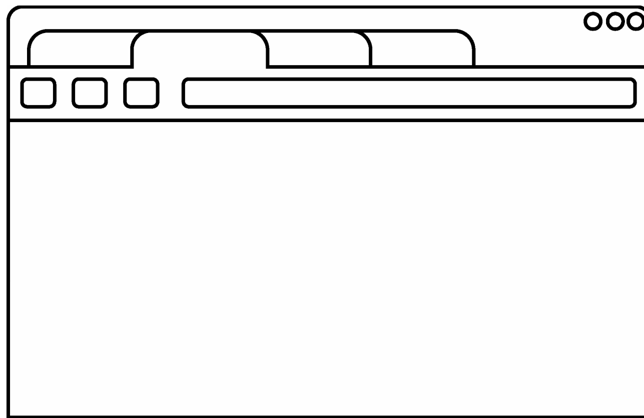
这样的设计让我每次都犯错误，我会认为当前播放内容的下一个编号是章节编号，而不是音轨文件编号。

iPhone 的 Audible 应用解决了这一问题。它完全不显示音轨文件编号，而把章节编号放在了当前播放内容的后面，这样就修正了容易让人产生误解的布局。



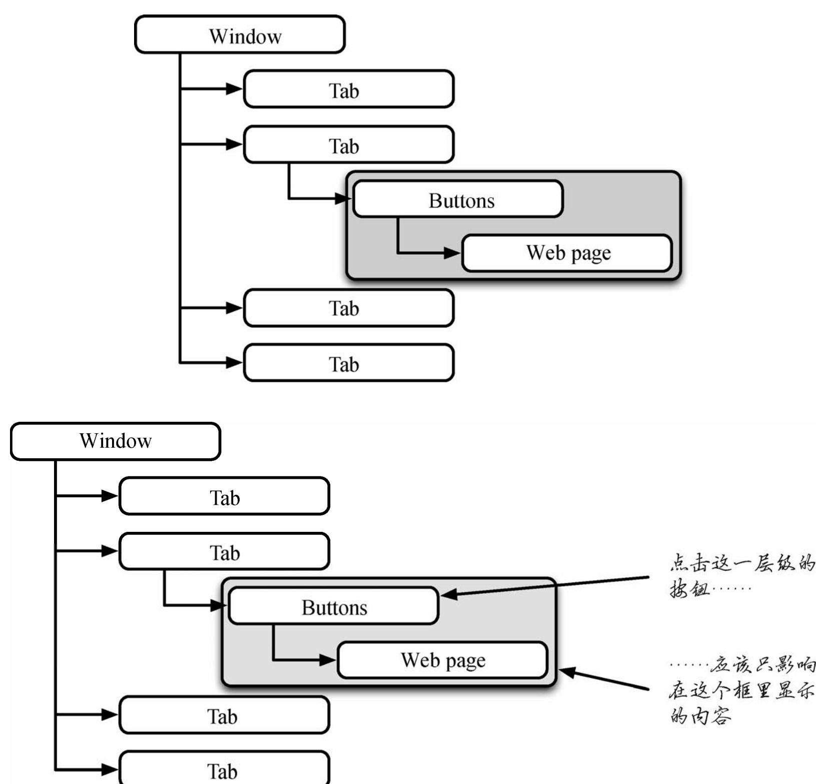
人们通常认为屏幕上显示的有层级结构关系的界面元素应该从上到下排列。在设计屏幕布局时，一定要保证界面的视觉层级结构和各个界面元素的层级结构一致。

让我们返回到浏览器的例子。如果只看其主要界面元素，Chrome 的用户界面如下所示。



当人们看到这幅图像时，他们凭直觉可能会认为这个窗口中的元素层级结构如下页第一幅图所示。

这种想象的层级结构可以让人们明白用户界面的行为方式：关闭某个标签会让这一层级下的所有东西都消失。点击按钮栏上的按钮会影响到位于同一层级或下一层级的用户界面。



与某个用户界面元素进行交互的时候，不应该影响上一层级内显示的内容。

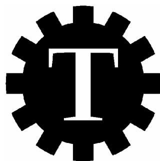
层级结构无处不在，它们影响着人们对产品行为方式的预期。适当地使用层级结构可以给用户一些提示，让用户知道该如何使用产品。

提示

- 考虑如何布局产品元素的层级结构。
- 使用层级结构向用户提示产品的工作方式。产品的各个界面应该向用户显示隐性或显性的层级结构。

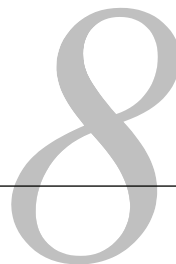
延伸阅读

设计师把那些大部分人称为“组织事物”的行为称为信息架构（information architecture）。关于这一主题有很多不错的书籍，如唐娜·斯宾塞写的《信息架构实用指南》（*A Practical Guide to Information Architecture*）。另外一本比较好的入门类书籍是皮特·摩威利和路易斯·罗森菲尔德合著的《Web 信息架构》（*Information Architecture for the World Wide Web*）。



第 8 章

卡片分类





什么是卡片分类？

人们通常会认为产品的某些部分应该出现在某些特定的位置，某些内容应该显示在一起，卡片分类可以帮助我们收集这类信息。

如果产品比较复杂，必须对其内容进行分类，那么可以考虑使用卡片分类技术。如果你正在设计一个网站，需要决定如何组织各个页面，或者你正在开发具有多种功能的应用程序，要以用户可以理解的方式对其功能进行组织分类，那么本章内容正是为你量身定做的。

为什么要使用这项技术？

我们和用户对于产品的看法是截然不同的。比如，假设我们在为某个企业创建网站，如果我们已经非常了解企业的内部组织形式，就会影响到我们设计网站的组织结构，但访问网站的人可能根本不知道企业内部是什么样的。这就意味着，对我们设计师来说非常好的东西，可能用户根本无法理解。

卡片分类可以帮助我们找出人们对事物的看法。

是否存在前提条件？

不存在。

8.1　设计层级结构

在上一章中我们了解到，产品通常是用层级结构组织起来的，让我们来看一个例子。

随便打开一个网站，它们可能都是通过层级结构组织起来的。假设你的笔记本的内置摄像头没法用了，你需要帮助。首先在浏览器内打开制造商的网站，在哪里才能找到帮助信息呢？如果你的电脑来自于较大的生产厂商，你可以先点击“电脑”，然后再点击“笔记本”，之后就能看到

制造商生产的笔记本电脑系列清单，点击正确的产品系列，能看到一系列笔记本电脑图片，你应该能从中找到自己的那款。在具体产品的页面内，你可以找到“技术支持”链接。理想状况下，点击这个链接之后就能看到自己那款电脑的技术支持页面，其中列出了电脑可能出现的一系列问题。如果够幸运的话，要解决的问题和解决方法就在这个页面里。

以下是你在整个流程中经过的网站层级结构：

电脑→笔记本→笔记本产品系列→笔记本图片→笔记本技术支持→具体问题的答案

但这并不是到达最终答案的唯一路径。你也可以直接进入网站的“技术支持”版块，寻找问题的解决方案。问题是如果你正在设计这个网站，你怎么知道用户会走哪条路？你怎么知道用户期望看到网站的哪些页面？

卡片分类

解决这些问题的常用方法是使用卡片分类，这是找出人们期望事物所在位置的绝佳办法。卡片分类非常容易。唐娜·斯宾塞在其著作《卡片分类：可用类别设计》（*Card Sorting: Designing Usable Categories*）中描述道：“卡片分类是一种非常简单的技术，只需在带有编号的卡片上写下一些内容，然后让人们对这些卡片进行分组就可以了。”

当然还有一些实施细节，下面介绍卡片分类的细节问题。



8.2 准备工作

在进行卡片分类之前，你只需准备一些带有编号的卡片，然后在上面写下要进行分类的内容。例如，如果你正在设计网站结构，那么就在卡片上写下网站各个页面的名称，或是网站包含的版块；如果你正在设计应用程序，那么可以在卡片上写出产品的特征、功能、菜单项、命令、窗口、

任务、目标或是视觉元素的名称等。

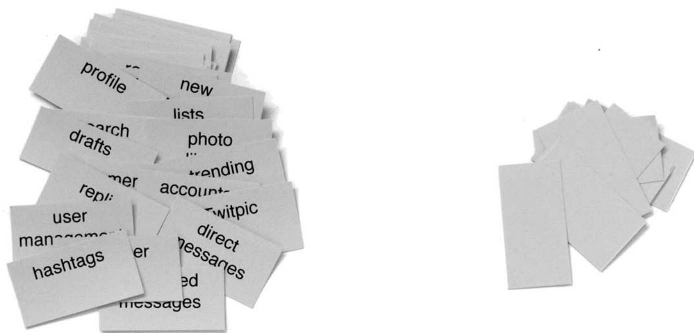
有时候，你要选取产品中位于同一层级的东西作为卡片的内容，不要把相差很远的层级内的东西混在一起。你应该为不同层级的信息进行多次卡片分类。

卡片上词语的含义最好显而易见，易于理解。如果你实在无法避免要使用一些行话，那么在进行卡片分类前要解释清楚这些词语的含义，并允许参与者用自己的词汇代替这些词语。

在选择卡片分类中要用到的词语时，你要确定人们不会因为某些词语的外在相似性而将其分在一组，例如某些词的词形相似或是发音相近。如果不考虑这一点，那么参与者很可能会因此把这些词分在一组，根本不去思考其真正的含义。[①]

卡片总数应该控制在 20 到 80 张之间。如果少于 20 张，那么你思考问题可能还不够深刻；如果多于 80 张，可能会让参与者受不了。

另外，你还要准备一些空白卡片。



因为人们可能会在卡片分类过程中写下一些东西或是涂改卡片内容，而且你要进行多次卡片分类，手工制作卡片非常耗时，所以，简单的做法就是把卡片打印出来，然后把它裁剪好。

8.3　参与人员

你可以每次只让一个参与者进行卡片分类，也可以让多个参与者同时进行。这两种方法各有优劣。每次让多个参与者共同进行卡片分类比较难以协调进度，如果是一个人的话就不存在这个问题。但是多个参与者可能会相互交流，激发出很多有价值的见解。

如果同时让多个参与者进行卡片分类，一定要控制人数。每次执行卡片分类的参与者不能超过 3 ~ 4 个人。

---

① 雅各布・尼尔森也曾讨论过这一问题，并且给出了一些解决方案，参见 http://www.useit.com/alertbox/word-matching.html。

一共要进行多少次卡片分类呢？雅各布·尼尔森给出的建议是，[1]每次用户研究最好进行 15 次卡片分类。卡片分类非常简单，多次进行是轻而易举的。不同的人对待事情的态度不同，你进行卡片分类的次数越多，问题的答案就越清晰。

8.4 执行卡片分类

向参与者致过欢迎词之后，要向他们说明要做什么事情。你可以这样介绍：

> 嗨！大家好，我是卢卡斯·马西斯。我最近正在设计开发一款新的 Twitter 应用。Twitter 是一个类似于 Facebook 的社交网站。
>
> 现在，我正在构建该应用的基本架构，也就是决定应该在什么地方呈现什么样的功能之类的事情。我们即将做的事情能够帮助我了解人们对该应用的期望。
>
> 等一会我会发给你们一些带有编号的卡片，上面写了一些词语。这些词描述的正是应用具有的一些内容，比如功能或目标之类的。我希望你们做的是按照你们自己的理解，把你认为应该放在一起的卡片分在一组。说得更明白一点，我们要做的不是寻找这些卡片表面的相似性，不是把那些发音相近或是首字母相同的词放在一起，而是假想你正在使用这款应用。比如，你希望在同一屏内看到哪些东西？如果屏幕上显示了一个列表项，你希望在这个列表内看到些什么？把你想看到的内容放在一组就行了。
>
> 如果有一些卡片你实在不知道应该放在哪一组，或者你认为它根本不应该属于这款软件，也不要太过纠结，把这些卡片单独分成一组就行了。我手头还有一些空白卡片，桌子上也有马克笔，如果你不理解卡片上某个词语的意思，就直接把它划掉，然后换一个自己认为更好的词。如果你认为某些卡片应该分属于多个不同的组，你也可以自己复制这些卡片。
>
> 在开始之前，让我们先浏览一下这些卡片。

然后和参与者共同浏览卡片，确保参与者能理解每个卡片的含义。

接着请参与者根据自己对卡片的理解进行分类，让他们把分在一组的卡片堆在一起。



[1] 相关文章参见http://www.useit.com/alertbox/20040719.html。

请参与者大声说出自己的想法，并让他们提问相关问题，此时你要记笔记。如果参与者提出了一些新词语，你可以为其制作额外的卡片，或者把这些新词语添加到相关的现有卡片中。

如果参与者分出的组中只有很少的卡片，鼓励他们把相似的组合并在一起；如果他们分出的组内有很多卡片，鼓励他们进行更细的划分。

根据实际情况，你还可以执行“封闭式”卡片分类：事先定义一些组，然后让参与者把卡片分到不同的组内。

让参与者不要对那些他们认为放在哪里都不合适的卡片进行分组，这一点非常重要。有被放弃的卡片表示产品的某些功能或网站的某些版块并不十分重要，或者这些东西与用户通常使用产品解决的问题没有任何关系。

在分组完成后，让参与者为每一个卡片组命名。在每一叠卡片的上方添加一个卡片，写上该组的名称（通常用不同颜色的卡片）。



如果分出的组够多，请参与者对这些组进行排列，把他们认为有联系的组放在一起。根据参与者以及卡片上词语的数目，你甚至可以让参与者用线把相关的组连起来。（如果有意要这样做，你需要准备一张大纸来实施卡片分类。）

最后，收集数据。一个快速简便的方法是用相机拍下整个布局，然后把每个卡片组用橡胶圈固定好。

8.5　远程卡片分类

你不一定非要邀请人们聚集到一起来进行卡片分类，通过远程方式也很容易进行。实际上有很多比较好的卡片分类网站，比如 websort.net 和 OptimalSort[1]都是专门用于远程卡片分类的网站。

---

① 分别参见 http://uxpunk.com/websort/和 http://www.optimalworkshop.com/optimalsort.htm。

远程卡片分类能够收集到更多的数据，因为你可以让更多的人参与，但也将损失一些聚在一起能够获得的有效信息。比如，哪些词语让参与者迷惑不解？参与者使用了什么样的同义词？他们对卡片上的词语产生了什么样的联想？远程卡片分类无法为你带来这些信息。

当然，你完全可以同时用以上两种方式进行卡片分类。

8.6 评估结果

如果你在卡片分类之前已经预先设定了一些分组（也就是进行封闭式卡片分类），那么对结果进行评估是非常简单的。计算出某个卡片在每个组内出现的频率，你就能得出大部分人希望这个卡片所表示的内容出现在什么地方。

	组1	组2	组3	组4	组5	组6	组7
卡片 1		4		3			
卡片 2			2	6			
卡片 3	7						
卡片 4			5	1		1	
卡片 5			5	4			
卡片 6	1	4					3

相反，如果分组是由参与者定义的，那么你首先要重新定义这些分组。有时，不同的人会用不同的词汇描述同一个组，这时你就要把这些组合并起来。另外，不同的人可能会以完全不同的方式对卡片进行分类，创建出完全不同的组。如果出现了这种情况，那么你可能就要向用户提供不同的访问方式。在确定了要使用哪些组之后，再次计算每个卡片在所有组内出现的频率。

根据所收集的信息，你就可以构建出产品界面的层级结构了，然后将这一层级结构应用到产品的视觉布局和信息架构上。我不建议对收集的数据进行正式的统计学评估，因为你可能根本就没有收集到足够的信息。卡片分类是为设计决策提供支持的，而不是用来证明某个解决方案是“正确的”。

在为应用程序绘制故事板、构建网站内容或各个界面的层级结构的时候，只要把收集到的信息考虑在内就可以了。卡片分类能为你提供用户对产品的看法、用户对产品体现出的概念进行分类的标准、用户关于产品工作原理的心理模型等。

在确定了层级结构之后，通常需要创建一个简单的纸质原型，并进行数次简短的可用性测试。（关于如何开展可用性测试，参见第 11 章。）这样，你就可以得知自己对卡片分类所得结果的解析是否正确。

8.7　可用层级结构的创建准则

卡片分类收集到的资料可以帮你设计出优秀的层级结构。以下是帮助你完成这项设计的一些准则。

允许同一内容在多处出现

记住，层级结构的设计不必严格死板，某一内容不必局限于某个固定的位置。你可以为用户提供多种方式去访问同一内容。这叫做“多样化层级结构分类”。图 8-1 即是一例。

前面曾提到过，有些用户在电脑出问题后会首先搜索网站，寻找自己电脑的型号；而其他人则可能直接进入网站的“技术支持”版块来寻找自己电脑的型号；也有人可能会在进入“技术支持”版块之后根据具体的问题搜索解决办法。如果你发现不同的人在卡片分类过程中把卡片分成了不同的组，那么你就要创建一个宽松的层级结构。

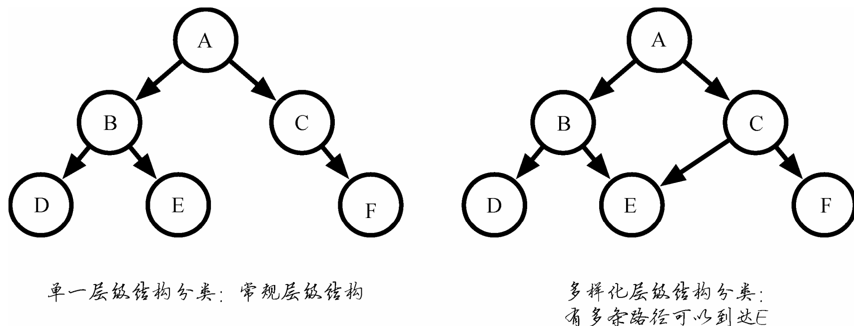


图 8-1　多样化层级结构分类示例

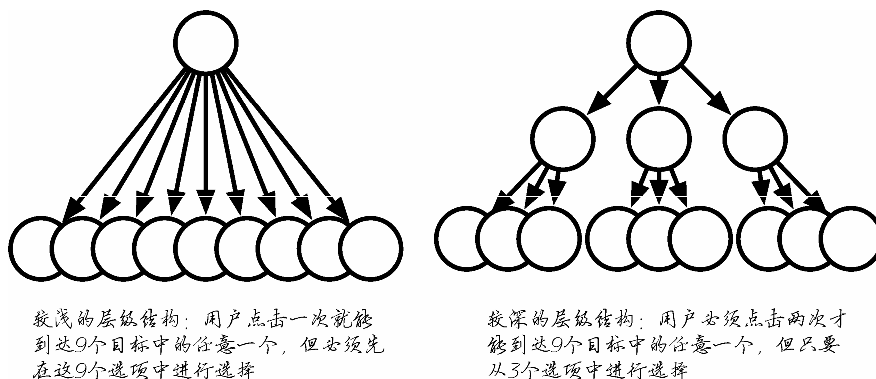
深和浅的问题

大部分时候，人们都是通过点击次数来评价用户界面的。用户应该点击多少次才能到达目标位置？凭人的直觉来讲，点击次数越少越好。

这种只关注点击次数的评价方法会引导你把层级结构做得尽可能地浅，以便让用户通过尽可能少的点击次数实现目标。虽然这样能够确保用户只通过少量的点击就能到达少数重要的、常用的功能，但我个人不建议你因此特意采用较浅的层级结构，因为这样会迫使用户在每一层级，都要从大量的可行操作中选择自己想要执行的操作。减少层级结构的深度会增加每一层级上可选择选项的数量。

实际研究表明，减少点击次数的优化并不一定必然产生好的结果。uxmyths.com 在总结研究成果时写道：“所需的点击次数不会影响用户的满意度，也不会影响用户的工作效率。事实的确如此，更少的点击次数并不能让用户高兴，也不是提升效率的必备条件。”①

① 参见 http://uxmyths.com/post/654026581/myth-all-pages-should-be-accessible-in-3-clicks。



与“点击次数越少越好”这一准则类似的是，你必须控制呈现给用户的选项数量，不要超过7个。得出这一结果的理论基础是：人类不能处理超过7个的可选项。与“更少点击数”一样，这一原则也是错误的。

“7项准则”来源于普林斯顿大学认知心理学学者乔治·A.米勒在1956年发表的一篇论文[①]。作者在论文中总结道，人类只能在工作记忆中记住7个不同的信息块。虽然这一观点可能是正确的，但你不能把这一准则应用于在多个选项中进行选择的过程。因为用户不必把所有的选项都记住，而只要浏览这些选项，选出第一个能让自己更接近目标的选项即可。这一行为过程称为“满意度”，心理学家赫伯特·西蒙在1956年提出了这一术语，用户并不是比较所有选项来寻找最佳答案，而是选择第一个看上去感到最满意的选项。

但是，巴里·施瓦兹却在《无从选择：为何多即是少》（*The Paradox of Choice*）中认为，有些人本身就是“完美主义者”，他们试图找出最好的方案，而不是选择第一个适合的选项。另外，虽然大部分人都可以从大量选项中进行选择，但很多人并不愿意这样做。施瓦兹写道：“大量的选项可能会打击到客户的积极性，因为这样会提升他们做出决策的难度。”

也就是说，即便某些人完全能够在众多选项中做出选择，他们也根本不愿意这样做。

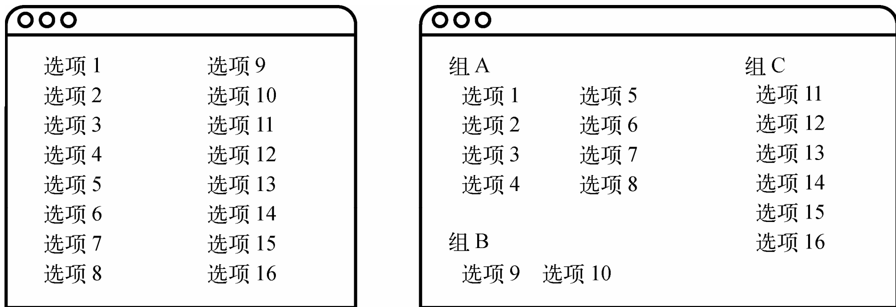
一个好的用户界面，并不是指用户通过少量点击就能到达目标，或者用户每次只需要从少量选项中进行选择，而是指每个点击都是显而易见的。如果用户拥有明确的目标，那么每个层级内都应该包含一个能满足其目标（至少也要接近目标）的选项。只要用户感到自己正在一步步接近目标，他们并不介意进入到一个较深的层级结构中。

关于分组

人类或许能够在多个选项中进行选择，但这并不意味着你可以把未经分组的选项扔给用户。面对着一大堆好像全都有效的选项是非常让人沮丧的。

① 维基百科有一个精彩的条目讨论这篇论文，参见 http://en.wikipedia.org/wiki/The_Magical_Number_Seven,_Plus_or_Minus_Two。

对选项进行合理分组或排序对用户很有帮助。良好的分组便于用户浏览大量可用的选项。



一次显示太多选项会让用户难以招架

把所有选项组织到局部的可视化层级结构中，这样做有助于用户对选项进行分组

使选项相互邻近并不是实现这种局部结构的唯一方法。史蒂芬 · E.帕默尔在其著作《视觉科学：从光子到现象学》（*Vision Science: Photons to Phenomenology*）中介绍了大量对元素进行分组的方法。以下是 8 个未经分组的圆点，无论怎么看，都看不出哪几个点是属于一组的。



但只要稍微调整一下这些点的位置，就可以让浏览者觉得这些点被分成了 4 组，而不是 8 个单独的点。正如刚才提到的，这种方法称为“邻近”。



另外一种把点分成两个一组的方式是：改变其中 4 个点的颜色，然后根据颜色把所有点分成 4 组。



改变点的大小也可以实现同样的效果。



或者改变其方向。



以上三种方法分别根据颜色、尺寸和方向对点进行分组，对事物进行分组不限于这几种方法，例如，通过使用不同的字体样式也一样能达到分组的目的。

另外一种更为明显的对元素进行分组的方法是把它们放进相同的区域内。



或者直接把两者连接起来。



还有很多种方法可以对事物进行结构化和分组。在一屏内显示大量内容时可以使用这些方法进行分组。

提示

- 不要把产品的内部结构作为层级结构的设计方案，在产品或网站上展现给用户。
- 用卡片分类找出人们关于事物工作原理的看法。
- 不要把减少点击次数作为优化整个层级结构的目标。
- 用分组法来规划各个屏内要显示的内容。

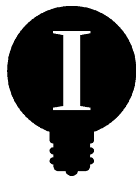
延伸阅读

只需要阅读一本书，就可以找到所有关于卡片分类的内容，那就是唐娜·斯宾塞的《卡片分类：可用类别设计》。

如果你对更广泛的信息架构领域感兴趣，皮特·摩威利和路易斯·罗森菲尔德合著的《Web 信息架构》和唐娜·斯宾塞的《信息架构实用指南》都是非常好的入门书籍。斯宾塞还有两个博客，分别关于卡片分类[①]和信息架构[②]。

如果对人们理解事物的方式感兴趣，那么精彩的《视觉科学：从光子到现象学》值得一读。

① 关于卡片分类的文章，参见 http://rosenfeldmedia.com/books/cardsorting/。
② 关于信息架构的文章，参见 http://practical-ia.com。



第 9 章

心理模型



使用产品的用户并不是中立的、毫无偏见的。他们在使用产品之前对产品应该如何工作就已经有了自己的观点。但大部分产品都没有按照人们所期望的方式工作，人们不得不“学习”如何使用产品。

与其强迫人们去学习如何使用产品，创建出按照人们的期望工作的产品不是更好吗？乔尔·斯波斯基在《程序员的用户界面设计指南》一书中说道：“只有某个程序完全按照人们所期望的方式工作，其用户界面才能称得上是优秀的设计。”

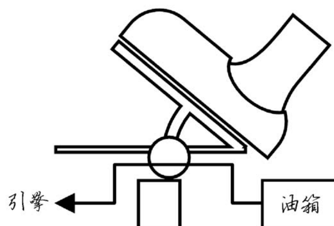
当然，不同的人对事物的工作方式持有不同的观点。但即便如此，你也可以努力缩小人和产品之间的鸿沟，让产品的实际工作方式更接近人的期望。

9.1 人的思维

用户对事物工作方式所持的观念称为“心理模型”（mental model）。它通常比实际情况简单。在《交互设计精髓》中，艾伦·库帕将心理模型描述为“一种认知速记，……，它足以描述用户和产品的交互过程，但不一定反映实际的交互机制。”

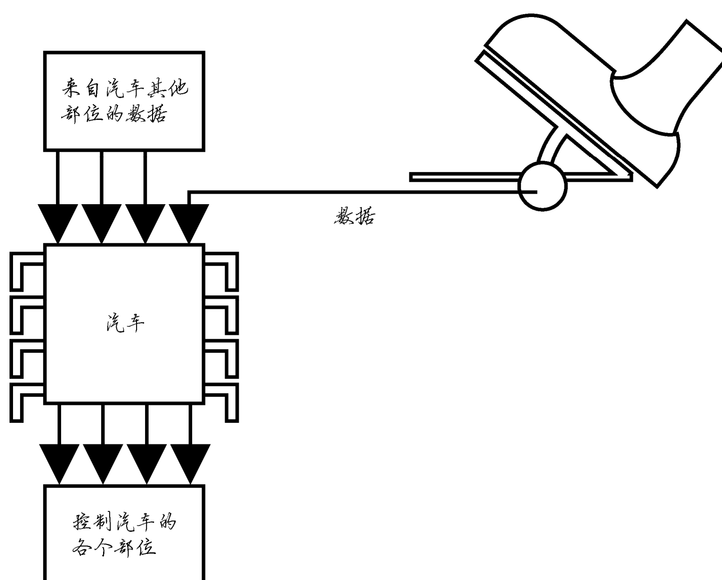
雅各布·尼尔森认为：“心理模型是用户对周围系统的认识。”[1]

让我们来看一个例子，某些开汽车的人认为油门踏板和引擎是通过机械结构直接连接的，认为踩下油门踏板能够增大某个值，让更多的燃料进入引擎，这样汽车就能跑得更快。



① 参见 http://www.useit.com/alertbox/mental-models.html。

这个关于汽车工作原理的心理模型实际上是错误的。其实，与油门踏板连接的是电脑，油门踏板输入的数据只是电脑所要考虑的众多数据中的一部分，燃料系统也只是电脑所控制的众多系统中的一个。



电脑基于所有这些数据分析驾驶者的意图：他是因为上了高速公路，所以要尽快提速行驶，还是因为红灯变成了绿灯，从停止状态启动汽车，想快速提升车速，又或者是因为想尽快地把车停住，突然间放开了油门踏板？

如果电脑能正确工作，驾驶者根本不会注意到分析过程。踩下油门踏板，汽车会更快，就像他们所希望的那样。

此处有一个重要的问题：从技术层面讲，用户关于汽车工作原理的心理模型是错误的，但这一模型却有助于人们理解如何控制汽车，因为心理模型的交互逻辑与汽车的反应非常匹配（至少大部分时候是这样的）——踩下油门踏板，汽车会跑得更快。

换句话说，用户的心理模型不一定必须正确，只要和产品的行为方式一致就行了。

9.2　三种不同的模型

我们的产品实际上反映出三种不同的模型。

- 用户大脑中的产品工作原理，也就是用户关于产品的心理模型。

- 产品在用户界面中呈现的方式，我称之为“UI 模型”。①
- 产品实现功能的过程，我称之为“实现模型”。②

在理想的产品中，以上三种模型应该是一致的。用户界面完美地表现产品实现功能的过程，用户能准确地理解他所看到的东西。

9.3 隐藏产品功能的实现细节

在现实中，以上三种模型常常并不一致。比如，实现模型太过复杂且不通用，所以你必须简化用户看到的东西。这与保持实现模型与 UI 模型的一致并不冲突。

拟人论（Anthropomorphism）

人们通常会把某些人类特征赋予应用程序、网站或其他产品。*这也是一种心理模型，人们认为机器的工作原理与人相似。因此，当某个数币机很快完成了任务时，我们会认为它是敷衍了事。我们会对电脑生气，好像它是有意“侵吞”我们的文档，不让我们找到似的。我们会认为汽车的导航系统故意把我们引入歧途。我们还会为设备取名字。

汽车制造商很会利用我们这种倾向来使他们的产品拟人化。他们有意把汽车设计得具有人的某些特征，让汽车看上去更友好或更具侵略性。



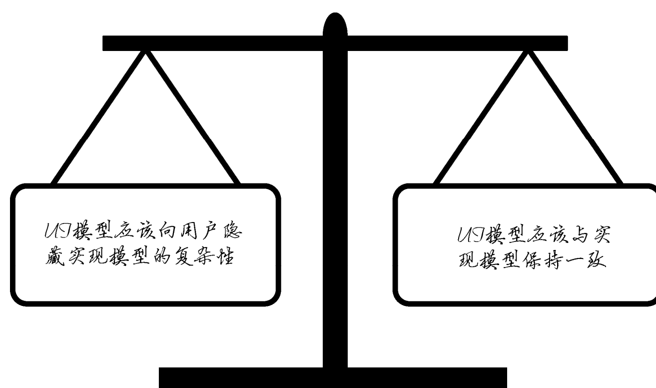


作为设计师，我们应该留意人们会从我们的产品中看出哪些人类特征。产品的行为方式像友好的人，还是不友好的人？它看上去是平静、淡漠，还是具有自己的个性？它是慢慢吞吞，还是效率高得让人起疑？

* 克里夫特·纳斯根据自己在这方面的研究写了一本名为《你会对你的电脑说谎吗》（*The Man Who Lied to His Laptop*）的书。

① 产品的外在表现（用户界面）形成的模型，有时被称为设计模型（design model）、表现模型（manifest model）或者设计师模型（designer’s model）。

② 产品功能的实现过程形成的模型，有时被称为系统模型（system model）或程序员模型（programmer’s model）。



这是个权衡过程。你要简化用户界面，这样大部分用户才能拥有美好的体验过程，但这意味着 UI 模型可能无法与实现模型完全保持一致。

假设你要创建一个在线平台，让浏览者购买可下载的电影。网站的潜在客户可能这样理解购买一部影片的过程。

(1) 进入商店。
(2) 浏览一大堆的电影，直到找到自己想要的。
(3) 为自己选择的 DVD 影片付钱。

这就是用户关于如何购买影片的心理模型。

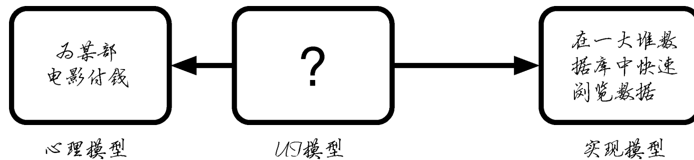
但这并不是在网上购买一部电影的过程，客户支付的并不是实物货币。实际情况是，服务器端的软件会向信用卡公司发出请求，在数据库中改变一些数字，然后更新数据库中的一些信息，告知客户可以下载某个文件了。当客户开始下载文件时，文件带有水印，可能还是加过密的。一旦文件下载完成，客户端电脑上的软件会对文件进行解密，解码所下载的电影，然后在客户的屏幕上播放。

我们该如何在用户界面中呈现这一系列复杂的过程呢？

你比用户知道得多

如果你正在为某个产品设计用户界面，你可能已经知道了产品功能的实现方式。如果你正在为某个企业创建网站，你可能已经了解了企业的内部结构。但用户很可能并不知道这些。在设计的初始阶段，你的心理模型和用户的心理模型就已经出现了分歧，因为你的心理模型建立在用户根本不知道的信息之上。

某些对你有用的东西不一定对用户有用！



关于如何购买电影，用户的心理模型和实现模型是完全不同的，而用户界面正好位于这两者之间。因此，用户界面的任务就是连通这两个不同的世界，将那些发生在用户视野之外的奇怪的事情以一种用户能够理解的、与之息息相关的方式呈现出来。为了帮助用户理解当前正在发生的事情，UI 模型必须比实现模型更接近于用户的心理模型，必须隐藏实现模型中某些复杂的过程。

9.4　抽象漏洞

对用户隐藏产品的实现细节会让 UI 模型更易于理解。但是当出现抽象漏洞（leaky abstraction），隐藏的实现细节被泄漏给用户时，用户会发现自己的心理模型无法与产品的行为方式相匹配。

让我们回到前面购买电影的例子，假设某人要为自己的妹妹在网上购买一部电影，她关于购买一部电影作为礼物的心理模型非常简单。

(1) 进入某个商店。
(2) 浏览大量电影，直到找到她妹妹喜欢的。
(3) 为挑选的 DVD 影片付钱。
(4) 用包装纸把 DVD 影片包起来。
(5) 把包装好的 DVD 影片送给她妹妹。

当此人在网上为她妹妹购买电影时，以上心理模型完全崩溃。UI 模型隐藏了产品实现模型的关键细节：用户购买的电影是加密的，需要对其解码后才能观看，并且只能在她自己的电脑上进行解码才能观看。用户认为她可以把下载的电影放在记忆棒里，然后送给她的妹妹，这是合情合理的，也与用户关于电影工作原理的心理模型相匹配。但实际上，她的妹妹根本无法观看该影片。

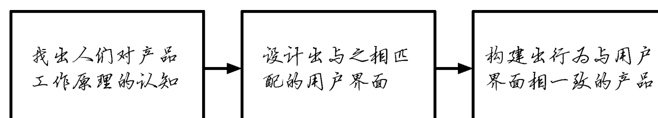
我们应该如何避免类似的问题呢？

9.5　为心理模型而设计

首先，你需要找出人们对事物工作原理的认知。通过与用户进行交谈，找出他们大脑中预先存在的心理模型（参见第 2 章）。然后进行卡片分类，找出用户对相匹配事物的认知（参见第 8 章）。再利用纸质原型进行可用性测试（参见第 11 章），这样才能做出符合人类心理模型的设计。向用户展示你的设计方案，描述一个交互过程，询问他们有何期待。（比如：“如果我键入一些文

字，然后关闭这个文档，你认为会发生什么？”）

利用从访谈和可用性测试中收集到的信息，你能够设计出与用户心理模型一致的 UI 模型。如果从草图开始创建产品，一定要确保产品的实现模型与 UI 模型一致。



这种理想状况并不总是能够实现。有时，你可能需要面对一个产品，它既无法按照人们的期望工作，又不能改变自身的工作方式。如果是这样，一定要保证用户需要学习的东西少而简单，只有这样，用户才能比较容易地改变关于产品的心理模型，适应产品的实际工作原理。

如果你发现用户在使用产品时（比如在可用性测试中）的心理模型总是错误的，那么可以尝试改变产品的外在表现。例如，不要使用那些与产品行为不匹配的隐喻（关于隐喻，参见第 12 章）。你也可尝试让产品看上去非常独特，这样用户将很快能理解为什么产品的行为与自己已知的东西不一致。

玛丽·乔·戴维森、劳拉·德芙和朱丽叶·韦尔兹在他们的论文《心理模型与可用性》（*Mental Models and Usability*）[①]中介绍了能够帮助用户形成有效心理模型的七个用户界面设计原则。

9

原则一：简单

心理模型是简化版的现实世界。如果你的产品遵从一些简单的原则，那么用户的心理模型很可能会与系统的实际工作原理相一致，用户也能更容易地掌握这些原则。

以 Flip 视频摄像机[②]为例，它背部有一个非常大的红色按钮。即便是从来没有使用过的人，只要看到它的设计就能形成正确的心理模型，知道该摄像机的工作原理：按下这个按钮就开始录制视频，再次按下它就停止录制。



（承蒙 Cisco 惠允）

① 参见 http://www.lauradove.info/reports/mental%20models.htm。
② 更多关于 Flip 的信息，参见 http://www.theflip.com。

原则二：熟悉

用户在使用产品之前已经具备了大量的经验知识。你的产品应该与相似产品保持一致，与现实世界中产品的工作原理保持一致，这样，用户才有可能形成正确的心理模型。

比如人们在当前大部分操作系统中删除文件的操作是这样的：把要删除的文件扔进垃圾箱，然后清空垃圾箱。你不需要向人们解释如何分辨空的垃圾箱和有文件的垃圾箱，他们的经验知识足以分辨出这两者的差别。



Windows 7系统中空的垃圾箱



有文件的垃圾箱

当然，这种关于垃圾箱的经验知识也会引发一些问题。在 Mac OS X 系统中，把 DVD 图标放进垃圾箱（这时垃圾箱会变成一个“弹出”符号）就会弹出 DVD。可以预见的是，很多人会因此而迷惑不解，毕竟人们不想因为这样的操作而损坏 DVD。

苹果 iPad 上的很多音乐应用程序都利用了这一原则。图 9-1 所示是一个名为 djay 的应用程序[①]。



图 9-1　应用程序 djay 的界面

① 更多信息参见 http://www.algoriddim.com。

它看上去非常像现实中的 DJ 转盘系统。这样设计的结果是，人们非常熟悉这样的系统，不需要学习任何东西就可以直接使用这个应用程序。因为用户已经具备了一个心理模型，该模型使他们可以很好地理解所看到的东西。

原则三：易于识别

不要让用户回忆如何做某些事情，而是要为他们提供一些有助于理解当前选项的提示或显而易见的选择。

以 Windows 7 系统中的菜单栏为例，它显示了与用户当前所浏览文件相关的操作。如果用户正在浏览图片，那么它就呈现与分享或浏览图片相关的命令，如图 9-2 所示。



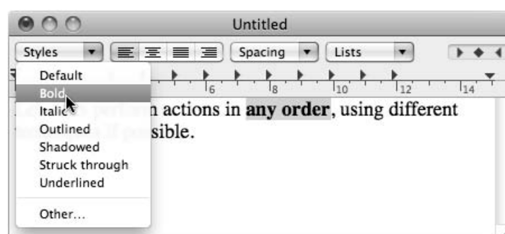
图 9-2 Windows 系统的菜单对情境非常敏感

用户不必记住如何使用幻灯片展示照片，当他们想要使用的时候，按键就已经随时待命了。

原则四：灵活

如果可以，允许用户以任意的顺序、使用不同的技术执行操作。

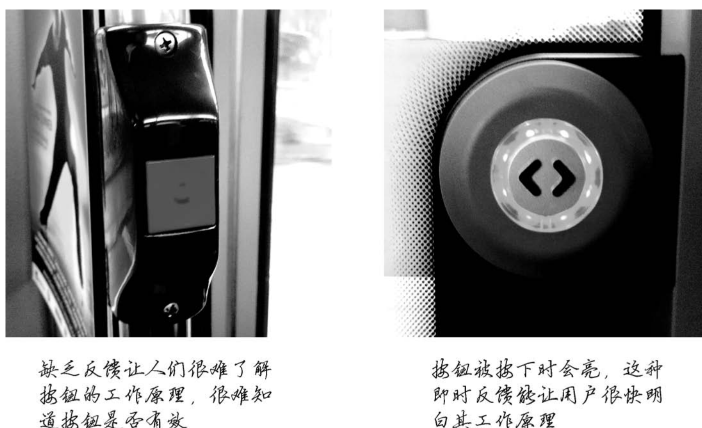
在现代的字处理器中，用户可以在键入内容的同时为文档的不同部分使用不同的样式。他们也可以先完成文字输入工作，然后再在文档中为各部分设置不同的样式。



原则五：反馈

要经常为用户的交互动作提供即时、有效的反馈。如果用户点击了某些东西，那么这些东西应该立刻以高亮效果显示。如果他们拖动了某些东西，那么这些东西应该随着鼠标或手指的运动轨迹移动，不应该存在任何延迟。如果他们启动了某个需要花一些时间才能完成的动作，就应该向用户显示一个动态指示器，比如用进度条来告知用户，电脑已经接收到了他发出的指令，正在处理（更多内容，参见第 23 章）。

图 9-3 所示的两张图片是苏黎士有轨电车上采用的不同按钮。当你想让电车停下时，按一下按钮就可以了。右边按钮的灯在按下时会变亮，而左边的则不会。在苏黎士乘坐有轨电车时，你常常会发现人们不停地按下左边这种按钮，只为了确认他们的按下动作已经生效，因为产品没有提供明显的反馈信息。而右边这种按钮则很少发生这种情况。



缺乏反馈让人们很难了解按钮的工作原理，很难知道按钮是否有效

按钮被按下时会亮，这种即时反馈能让用户很快明白其工作原理

图 9-3 按下这些按钮能让电车停下

当然，反馈过度也是有可能的。一些火车上的按钮采用了不同的色彩方案，某种颜色表示当前按钮可以按下（只在火车停下时短暂激活），另外一种颜色表示当前按钮已经被按下。由于人们根本无法记住这两种情况，他们往往就会重复按几次按钮，仅仅为了确认这个按钮转换到了正确的状态。为产品加入一些反馈机制能够帮助人们形成正确的心理模型。但如果加入的反馈机制过度或不清晰，会让人们更加迷惑。

原则六：安全

用户的操作应该是无害的，除非用户成心想做一些破坏性的事情。在关闭一个未保存更改的文档时，在损坏已更改的内容之前应该为用户提供反悔的机会，允许用户撤销他们的行为（参见第 19 章）。让用户可以毫无后顾之忧地自由探索，这样他们才能更加放心大胆地学习产品的使用方法，让自己的心理模型适应产品的工作原理。

例如，某人关闭了包含有多个标签页的 Google Chrome 窗口，但万一想要恢复某些标签页，

那么他可以通过历史记录菜单来恢复已关闭的标签页（见下页图）。

假想的因果关系

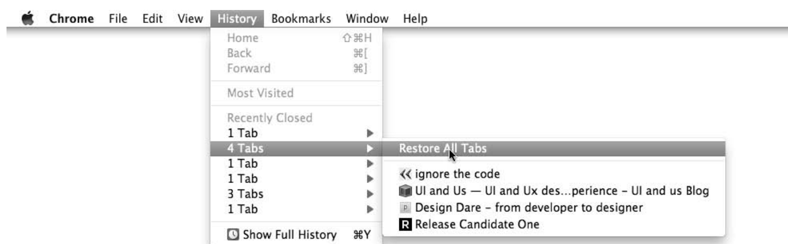
人们会假设那些同时发生，或者在临近时间内发生的事件之间存在一定的因果关系。如果用户执行了 A，马上就出现了 B，那么他通常会断定是 A 引发了 B。虽然你可以利用这种因果关系，帮助人们形成正确的心理模型，但也要记住，这样做也可能会适得其反。

这种问题通常发生在检测并维修故障的过程中。例如，某些东西不起作用了，用户执行了一些操作，尝试修复问题。如果这些东西又再次生效，那么用户自然会认为他的操作发挥了作用，毕竟，用户不知道如果自己不做任何事情，坏掉的东西是否会自己修复。

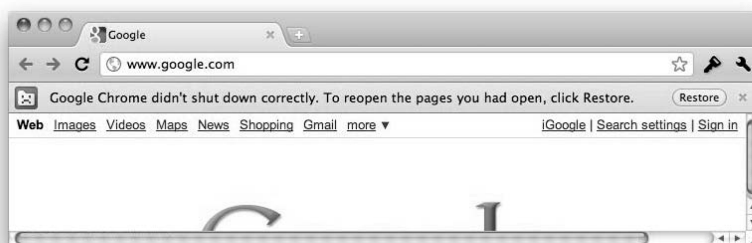
在瑞士，你经常会看到人们用硬币刻划投币设备，因为在第一次投币时，机器没有接收。



通常，机器在遭到刻划之后会接收硬币，因此，人们自然会想到：刻划机器会解决问题。但事实上，即便不刻划机器，再尝试一次也可以投进。



同样，如果 Chrome 崩溃了，它重新启动后仍然能够恢复在之前打开的所有标签页。



原则七：功能可见性

用户界面元素的设计应该向用户提示界面元素的交互方式。传达可行交互方式的设计细节称为功能可见性（affordance）。

唐纳德·诺曼在《设计心理学》（*The Design of Everyday Things*）一书中写道：

> 好的设计师会让用户看到适当的操作，而把不适当的操作隐藏起来。

任天堂的老式 NES 控制器（下图左）的设计没有体现出其把持方式。实际上，两个红色按钮的位置离右下方这么近，把控制器颠倒过来拿反而更舒服一些。对比任天堂的新产品 Game-Cube 的控制器，虽然新的控制器更复杂一些，但它很明显地告知了用户使用方式，控制器两边的把柄告诉用户应该用手握住它。



用户界面通常用斜面和高亮效果来显示那些用户可与之交互的元素。



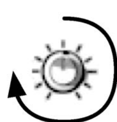
把鼠标移到某个界面元素的上方，如果它高亮显示，那么就表明你可以点击它。



当然，如果这种“鼠标悬停效果”使用不当的话，功能可见性也会对用户产生误导。如果为那些没有交互作用的元素使用了这一效果，用户就会误以为可以与这些元素进行交互。例如在 Mac OS X 系统中，甚至在用户把鼠标指针移到某个可点击的区域之前，窗口标题栏上的按钮就已经以高亮效果显示，这会让用户过早地点击某些区域，从而错过目标。



类似可能误导的功能可见性也常常出现在一些音频软件中，比如 Propellerhead 软件公司的 ReBirth。此类软件通常使用类似真实世界中的旋钮的设计元素，这样用户以为通过旋转能够与其进行交互。



界面元素提示的交互方式



实际交互方式

但事实上，你只能通过垂直拖拽的方式改变其值。

我将在第 12 章继续讨论应该如何把功能可见性应用于用户界面设计。

提示

- ❑ 人类对事物的工作原理有自己的观点。你的产品越接近这些观点，人们在使用产品时要学习的东西就越少。
- ❑ 为了匹配用户的心理模型，你必须经常向用户隐藏产品的实现细节。要小心提防那些抽象漏洞。
- ❑ 为心理模型而设计，要把产品设计得简单、熟悉、灵活并且安全。向用户提供反馈，并且在任何特定的时刻，都要明确地显示用户可选择的选项。

延伸阅读

艾伦·库帕的《交互设计精髓》有一章中准确章节很好地阐述了心理模型。罗伯特·霍克曼在《一目了然》一书中也曾讨论过心理模型。乔尔·斯波尔斯基的《程序员的用户界面设计指南》从开发者的角度阐述了心理模型。

雅各·布尼尔森曾写过关于心理模型的文章。[1]

① 参见 http://www.useit.com/alertbox/mental-models.html。

Part 2 第二部分

设　　计

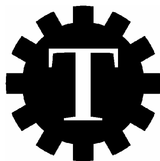
到目前为止，你已经明确了产品的用户对象以及所要解决的问题，或许也已经有了初步的解决办法。但要注意的是，此时还不能确定任何具体的设计方案。人类的大脑非常擅长对与已有观点相左的信息加以合理化，因此不要太早就确定设计意向。不要执着于某个特定的设计或解决方案，这样才能寻得更好的解决方法。

逐步地完善你的想法，不能操之过急。本部分的内容简单易行，但都是高招，可以帮你想出可行的产品工作方式。你要在设计过程中逐步加入细节，从制作产品的工作流程图，到制作故事板，再到简单的草绘图，最后是完整的实体模型，甚至是有交互功能的产品原型。

你可能会犯错，因此，一定要勇于承认自己的错误。执着于一个无效设计方案的时间越长，你就越难改变它。尽早承认失败未尝不是一件好事。

为确保自己的路线正确，也为了让自己的错误尽量出现在前期，就要让真正的用户对设计方案进行测试。很快你就会发现什么东西有效，什么东西无效。

本部分所介绍的内容在实际中是迭代进行的。不要太过主观。先设计，再测试，如果某些东西无效就承认它的错误，然后迭代整个设计和测试过程。



第 10 章

草绘与原型

10



草绘和原型分别是什么样的技术？

至此，你可能非常清楚自己想要创作什么样的产品。现在是时候进入详细设计阶段了，首先草绘出产品的结构，然后设计各个屏内所要显示的内容，逐渐进入细节设计。

为什么要这样做？

一旦设计方案开始付诸实施，再做出设计变更的代价是很高的。用户界面的小小变更也会对产品产生巨大的影响。

而在草绘图上做变更会非常迅速且代价很小。你所需要的只是橡皮、铅笔和几秒钟时间而已。

通常，你在草绘过程中所实现的只是产品最简单的原型。比如设计遥控器，你不可能直接创建模具，然后就进行生产，而是会先用木材或泥土制作遥控器模型，并通过模型来感受、调整遥控器的整体形态，然后再不断加入细节，直到实现最终想要的设计效果。

草绘图就像是你所创建产品的粘土原型。在开始编写代码之前，要尽可能多地确定产品细节。

是否存在前提条件？

你需要对产品的最终模样有明确的想法。

10.1　产品结构设计

在《重来：更为简单有效的商业思维》一书中，杰森·弗瑞德和大卫·海涅梅尔·汉森写道："在确定最终的建筑平面图之前，建筑设计师根本不考虑淋浴间该贴什么样的瓷砖，厨房里该装哪个品牌的洗碗机。"

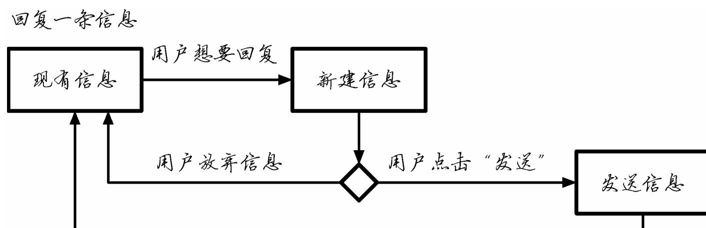
流程图和故事板就是你所创作产品的"平面图"。我们现在处于设计过程的最初阶段。流程图和故事板与细节无关，只关注产品的整体情况——结构。

10.2 流程图

流程图解决以下问题：用户要如何操作才能得到自己想要的东西？要按照什么样的步骤执行操作才能实现目标？

选择最为重要的用户目标，然后考虑实现这些目标所需的步骤。

对于我们的 Twitter 应用程序示例来说，回复一条信息的简要流程如下所示：



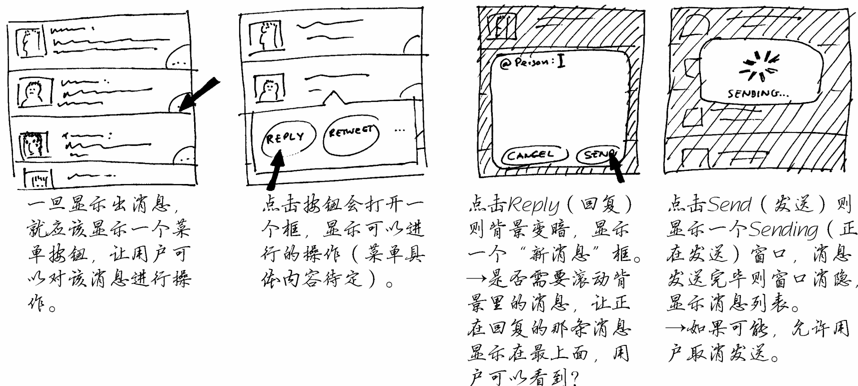
为流程图加上分支是可以的，但不要把流程图弄得太过复杂。从理论上讲，你可能需要用一个庞大的流程图才能表示整个产品。但在实践中，为最重要的几个用户目标创建多个流程图会更合理。这样，你才能将各个流程图布局得简洁明了。

这个练习的目标是考虑实现每个用户目标所需涉及的内容。你需要为用户呈现什么样的界面？用户需要在何时做出什么样的决定？

10.3 故事板

绘制故事板这项技术最初用来为电影规划动作场景。它可以把连续的动作分解成几个重要的帧，把一部电影变成一本连环画册。在用户界面设计中，我们通过绘制故事板来实现类似的效果，把用户路径分解成一系列的快照图片。

故事板通常会忽略分支，而仅关注交互细节。用户究竟在每一屏内都看到了些什么东西？你想让用户做什么？你需要在什么地方使用动画或图像来帮助用户理解他将要做的事情？



如果你设计的是在触摸屏上显示的用户界面，绘制箭头或手的形状向用户暗示能够在什么地方与产品进行交互，以及如何进行交互。

绘制故事板会花费很多时间，因此最好只针对应用程序中那些设计细节不明显的部分使用此项技术。每一个熟悉连环画册的人都明白什么是故事板——它是一款交流设计的绝佳工具。如果你需要向一个程序员解释如何实现某些东西，故事板可以提供很大帮助。

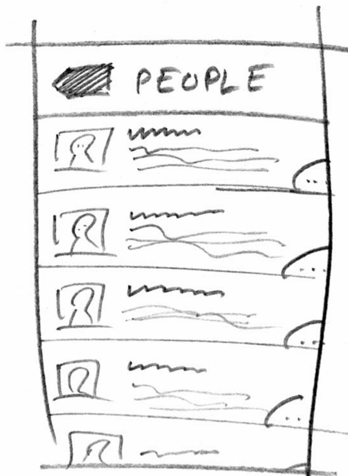
10.4　草绘

建筑师在设计完建筑平面图之后，就开始设计各个独立的房间。同样，在完成产品的结构设计之后，你就要开始设计各个屏幕显示的内容了。通过绘制流程图和故事板，你应该已经明确了所要设计的产品需要哪些界面，每个界面需要提供什么样的功能。

在绘制故事板的时候，你已经完成了某些界面的简单草图。但是，在故事板部分的示例中，有些屏显示的内容中含有与故事板所描绘的任务无关的元素。例如，弹出菜单不仅有一个“reply”按钮，还有一个“retweet”按钮。因此，现在我们需要确定各个屏幕中要显示的内容。

可用性咨询师布鲁斯・托格纳兹尼写道：[1]

> 一个开发小组所能犯的最糟糕的错误莫过于，立刻开始设计复杂的、完美的产品原型。……如果模型看上去不太完美，用户（或客户）就可以（非常自由地）发表不同意见。同时另一方面，如果设计师和开发人员没有把大量时间和精力挥霍在制作故事板或是临时使用的产品原型上，他们会更加愿意倾听客户的相反意见。



① 参见 http://www.asktog.com/columns/005roughsketches.html。

用简单的草绘图说明各个屏幕显示内容的基本效果。任何人都能绘制草图，因此你可以随意地邀请其他人加入到设计过程中。向他们展示你的想法，并倾听他们的建议。

只要你对所有屏幕显示内容的初步设计感到满意，就可以绘制线框图了。

Lorem Ipsum

人们通常会在线框图中使用以 Lorem Ipsum 开头的一段拉丁文字来作为填充文本[1]。如果你没有其他更好的内容，使用填充文本是可以的。但如果有，最好还是使用用户将在实际中看到的文本来填充线框。这会让你看到用户界面的实际工作情况，能帮助你确定文本的尺寸（如果某些内容有特定长度的话，例如 Twitter 信息）。

10.5　线框图

线框图用来展示屏幕所显示内容的结构，但它没有任何修饰——没有颜色、阴影或图片。线框只关乎要显示的内容，比如，你想在屏内显示什么内容，在何处显示，多大尺寸，间距多少。图 10-1（左）是一个线框图示例。

此时也是开始制作界面拷贝的阶段了。你不必一步到位，但一定要考虑线框图中应该包括哪些文本，这些文本应该放在什么地方。

绘制线框图的目的是明确在每一屏内具体显示哪些内容，在何处显示。如果完成了这一任务，就可以进入润饰阶段了。

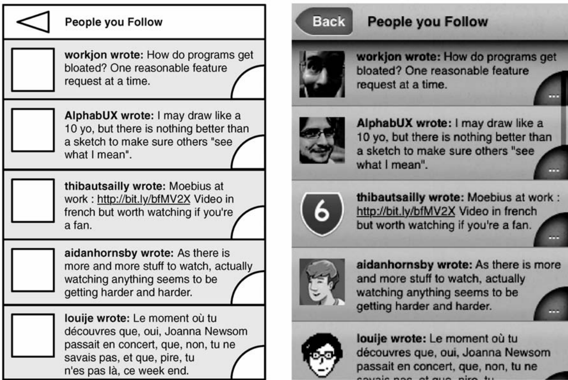


图 10-1　线框图（左）和实体模型（右）

① 参见阮一峰的博文《关于 Lorem ipsum》（http://www.ruanyifeng.com/blog/2009/04/lorem_ipsum.html）。——编者注

10.6　实体模型

线框图呈现的是屏幕布局，而实体模型则要在此基础上加入修饰或是视觉细节，如阴影、材质、图像、透明度等，也就是基于目前的知识，你希望界面呈现的效果。图 10-1 是实体模型（右）和线框图（左）的对比。

加入视觉细节并不仅仅是让产品看上去更加美观。当然，让最终产品更加美观确实是其目的，但视觉设计还可以向用户暗示产品的功能。用户界面设计师通常称之为“功能可见性”。我已在 9.5 节对此进行了详细阐述。下面就举一些功能可见性的例子：材质用来告诉用户他可以触摸或拖拽某些东西；阴影和斜面用来表示用户可以按下某些东西，或者用来表示层级结构；颜色用来传达目标的重要性，吸引用户的注意力。

在为产品的屏幕布局制作详细的实体模型时，你需要把以下内容谨记在心。

你不必用专门的图形处理程序来制作实体模型。如果喜欢，你可以用代码制作。在做出较大修改时，你尽可使用最顺手的工具。但要记住，现在制作的是产品原型，而不是最终产品的早期版本。安德鲁·亨特和大卫·托马斯在《程序员修炼之道》（*The Pragmatic Programmer*）一书中写道：

> 制作产品原型的目的是探索最终系统的具体特征。有了真正的产品原型之后，你将会抛开所有概念创作阶段的东西，吸取从中得到的教训，从而完善产品原型。

即便用代码制作实体模型，目标仍然是探索各种可能性，只不过这时删除那些无用的想法更为方便而已。

术语

本书中将出现以下专业术语：

草绘（Sketch）	用绘图表示用户界面的方式
线框图（Wireframe）	用户界面的一种静态表现形式，界面的各个元素都以设计尺寸显示在大致的位置
实体模型（Mock-up）	用户界面的一种表现形式（通常是静态的），其中加入了诸如阴影和颜色之类的修饰元素
原型（Prototype）	最终产品之前的所有产品的表现形式

并不是所有的人都像我这样定义这些术语。有时，“原型”一词仅表示有交互式功能的、高保真的产品表现形式。有时，“实体模型”一词用来表示最终用户界面的所有形式的草绘图。当你在网上或其他书中遇到这些词汇时，要注意作者表示的可能是其他含义。

10.7 工具

草绘和原型制作是软件开发团体中最常见的活动，以至于催生出了一个完整的产品开发生态系统。这个系统中的应用程序和服务存在的唯一价值就是帮助你绘制草图或创建产品原型。

使用类似软件产品的原因不胜枚举。其中一个主要原因就是这些软件可以让设计师们协同设计，即便他们相隔十万八千里。

Balsamiq[1]和 Mockingbird[2]是两个在线工具，你可以用它们来创建并分享自己绘制的用户界面草图。

Google Docs 的绘图模块称为 Google Drawings，它支持多人协同设计。Morten Just 上有一套 Google Drawings 的用户界面绘图模板[3]。

在Mac平台上，OmniGraffle[4]可以满足绘制用户界面草图的许多需求。人们已经专门为OmniGraffle创建了一些用户界面模板[5]。如果你是个 iPhone 开发者，还可以关注一下 Briefs[6]和 Review[7]。

一些人也用 PowerPoint[8]或 Keynote[9]之类的工具来创建实体模型和产品原型。这些工具甚至可以制作简单的动画和交互效果。Keynotopia[10]和 Keynote Kungfu[11]类的网站为这些软件提供了用户界面模板，能帮助你利用平台上的标准用户界面元素创建出无可挑剔的产品实体模型。

以上这些软件让产品开发变得更加简单。但使用纸张绘制产品原型仍然是非常有用的。它更快，更加原生态，并且只要所有人都在同一个房间里，就可以与其他人非常容易地协同工作，向其他人展示自己想法时只需要铅笔和纸而已。现在，基本上所有人都有附带内置摄像头的设备，向别人传递草绘的用户界面也变得非常容易，对着草绘图拍照，然后通过 email 发送就行了。

使用你感到最容易的工具。如果你擅长在纸上画草图，那么就用在纸上绘制草图；如果你更倾向于使用专门为用户界面设计师创建的软件，那么就用软件；如果你想用一款图像处理软件，那也无妨；或者，根据你当前要完成的任务结合使用所有这些工具。

① 参见 http://balsamiq.com。
② 参见 https://gomockingbird.com。
③ 参见 http://mortenjust.com/2010/04/19/a-wireframe-kit-for-google-drawings/。
④ 参见 http://www.omnigroup.com。
⑤ 参见 http://graffletopia.com/categories/user-interface。
⑥ 参见 http://giveabrief.com。
⑦ 参见 http://www.getreviewapp.com。
⑧ 参见 http://office.microsoft.com。
⑨ 参见 http://www.apple.com/iwork。
⑩ 参见 http://keynotopia.com。
⑪ 参见 http://keynotekungfu.com。

提示

- 首先创建产品的粗略图，然后逐步添加细节。先绘制流程图，再是故事板，然后是简单的草图，接下来是线框图，最终制作详细的实体模型。
- 尽早解决问题。你越早发现设计中的问题，处理起来所要付出的代价就越小。
- 流程图可以帮助你让用户以尽可能简单的方式实现他们的目标。
- 故事板可以帮助你为产品加入交互设计的细节，并就其进行沟通交流。
- 简单的草图能帮助你想出各个屏内所要显示的内容。
- 线框图可以帮助你决定在各个屏内的什么位置布局设计元素。
- 实体模型能让你进行快速的视觉设计迭代。

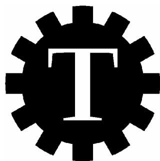
延伸阅读

流程图、故事板、线框图和实体模型是我最常用的四项技术，它们可以帮助我从产品的粗略图开始，逐步进入细节设计。然而，有些设计师家更偏向于使用其他技术。肯尼德・鲍尔斯和詹姆斯・鲍克斯的著作《潜移默化：用户体验设计行动指南》中有相关论述，该书还介绍了一些其他技术。

比尔・巴克斯顿的《用户体验草图设计：正确地设计，设计得正确》(*Sketching User Experiences*)也是这方面非常好的书籍。罗伯特・霍克曼在《一目了然：Web 和移动应用设计通识方法》中也谈到了这些技术。

泰勒・塔特也曾提出过一些绘制草绘和创建原型的其他方法。[1]

① 参见 http://uxbooth.com/blog/concerning-fidelity-and-design/。



第 11 章

纸质原型测试





什么是纸质原型测试？

前一章介绍了如何为产品绘制草图。我建议在编写代码之前完成这项工作，因为草图更容易更改。但是我跳过了这样一个问题：你如何才能知道该对产品做出什么样的修改？

有时候，产品草图一旦绘制出来，一些设计问题就会暴露无遗，比如，产品的某些功能总是出现于不恰当的时间或拥挤不堪的屏幕内。

但有时，一些问题并非那么显而易见。比如，哪种设计才是优良的设计？如果完成某件事情有好几种方案，哪种将会最有效？当前所采用方案的问题在哪里？本章将帮助你找出这些问题的答案。

草绘图是产品最基本、最原始的形态——原型（如果你喜欢这样称呼的话）。因此，你可以用它来进行可用性测试，看看设计方案是否像你设想的那样有效。最简单的办法就是把产品原型拿给用户，看看他们是否理解这些原型。

本章主要介绍如何对纸质原型进行完整的可用性测试。如果你独自开发产品或只是一个小团队的一员，略过本章内容也无妨。

为什么要这样做？

设计方案中存在的问题发现得越早，就越容易解决。你每用纸和笔修正一个问题，编写代码时需要修正的问题就会少一个。

是否存在前提条件？

你应该已经绘制了产品草图。

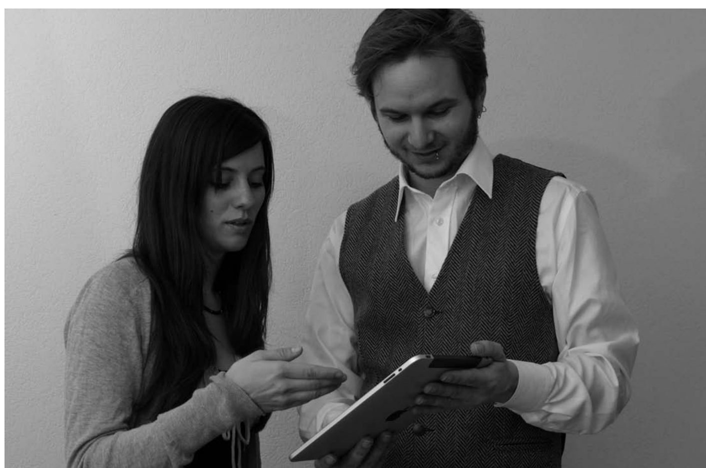
11.1　打游击式纸质原型测试

在这个阶段，你应该已经有了产品的静态原型。无论是简单粗糙的草绘图、线框图还是详细的实体模型，你的产品都已经以一系列图片的形式诞生了。

从最根本的层面上讲，纸质原型测试意味着你要找一个“真正的人”来与这些产品图片进行交互，从而判断你所规划出的用户界面能否为用户所理解。这非常简单，只要向别人展示某个用户界面的草绘图，然后问他们一些问题就可以了，比如：“如果你想更改这一屏显示文档的字体大小，你会点击哪里？”

要开展纸质原型测试，你需要准备产品某个界面的草图、线框图或实体模型，并且这些东西要足够详细。然后找一些愿意花 5 分钟进行测试的人，向他们展示这些产品界面，问他们一些常规问题，如：“你在这个屏幕内看到了什么？你认为这款产品是干什么用的？”

或者，你也可以问一些简单的、与任务有关的问题：“如果你必须用这个软件把一张图片添加到文档中，你会怎么做？”



纸质的原型？

本章介绍的是“纸质原型”，但“纸质”一词并不那么绝对。做这些测试，你不一定非要在纸上画草图，也可以用电脑制作产品的实体原型，然后将其打印出来，用它来进行测试。或者根本不打印，只是将其放在平板电脑上展示给参与测试的人。你甚至可以把某些草绘图放入 PowerPoint 或 Keynote 之类的软件，制作一些简单的交互原型来进行测试。

可能本章的标题更应该叫做“如何用最终产品的静态草图或实体模型（通常是这两种，有时也包括其他方式）进行可用性测试”，但这么长的名字与本书的布局设计实在是不搭调。

这种测试能让你大概了解用户是否理解你的设计，以及哪些部分用户不能理解。这样，你就知道自己在哪些方面做对了，应该对哪些地方进行修正。

只需向用户展示产品某一屏界面的实体原型，就能了解用户是否能理解这一屏内显示的东西。这样做已经很有效了，不过我们还可以更进一步，对一些交互方案进行测试。为此，你要预先准备好在多个屏内显示的内容。如果交互过程需要为界面加上或去掉弹出界面或其他元素，你也要预先准备好这些东西。在便利贴上画出这些内容，这样你就可以很容易地将其添加到产品原型上了。

做好了这些准备工作，你就可以进行任务可用性测试了。当然不能让参与测试的人为所欲为，偏离了测试工作。如果确实发生了类似的事情，那么说明你的设计方案存在问题，因为人们没有按照你所期望的方式进行操作。

你可以找任何人来执行这类简单的测试。测试时间很短，也很容易向参与者解释：

> 我正在开发一款新的 Twitter 应用。你知道 Twitter 吧？OK，很好。目前我正在进行这款应用的设计工作，我想知道用户是否能理解我已经设计出的东西。你知道，当一个人持续几周做同一件事情，他就无法对这件事做出客观的评价了。现在我就不知道自己所做的东西是否有用。我能否向你展示一些设计方案，问你几个问题呢？这样我就可以知道人们是否能理解我的设计了。这个测试不会超过 5 分钟的。

你可以先让自己的朋友、家人来做这种测试，然后再随机选择一些人进行测试。咖啡馆是个不错的地方，那里的人总有几分钟空闲时间。

一旦了解了人们如何理解你的设计方案，就很容易发现设计有哪些成效和潜在问题。

11.2　完整的可用性测试

之前介绍的简单的测试方法可用来对产品进行早期的可用性测试。当然，对产品原型进行测试还有很多细致的工作要做。对于大部分较小的团队，我不推荐使用完整的测试流程，但如果你有时间，并且能找到人参与测试，那么这种测试还是很有用的，甚至还没写代码就已经得到了很好的反馈。

典型的产品原型所展示的仅是最终产品的一小部分，一般无法包括最终产品的所有界面内容，也不一定能体现所有的产品功能。因此，产品原型的可用性测试总是基于某些特定任务进行的。分配给某人一项特定的任务，然后让他用产品原型完成这项任务。这样，你就知道应该在测试前准备哪些界面和 UI 元素了。

所以，你要做的第一件事情是确定测试的任务。

确定测试任务

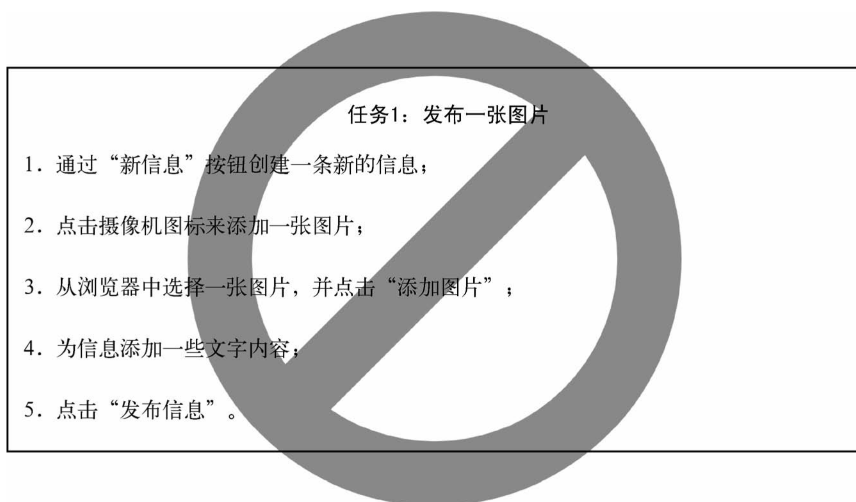
现在，你已经绘制了故事板，创建了实体模型，那么你应该知道了用户交互可能会在什么地

方发生问题。如果你很难为产品的某项功能设计出用户界面，那么就要尽早对其进行测试。如果这项功能是产品的关键功能、核心功能，那就更得这样做。选择那些对产品来说非常重要、你觉得可能会比较难用或者可能会存在问题的功能进行测试。

然后，制定出使用产品这些功能的任务。如果需要的话，你还可以利用早期进行用户研究得到的信息来制定测试任务。

记住，你并不是在寻求建议，可用性测试不是焦点小组访谈，其目的是观察测试者与设计方案进行交互的过程，从而找出交互设计存在的缺陷。一定要选一些能够让测试者真正使用产品的任务。

所选的任务不能有太强的指示性。告诉用户他们要完成什么目标，而不是要遵循什么步骤。下列任务是不恰当的。



这样的测试只会告诉你测试者是否能够遵守指令，而无法反馈出产品的可用性如何。对任务的描述应该是明确所要完成的目标，而把具体执行什么步骤留给测试者，比如：

> 你正在参加公司的一个客户活动。你想给房间拍个照片发布到公司的Twitter上。

这是用户可能会遇到的一个情境。它能让你看到处于这种情境下的用户如何与产品进行交互。

对任务的描述应避免使用官方术语，这听上去可能有点不对劲。但如果在任务描述中使用了用户界面中的词汇，实际上是在通过这些词汇暗示用户该如何完成任务。

至少要准备五至六项任务，越多越好，每项任务的持续时间应该为两到十分钟。不是所有的任务都能顺利完成，在实际开展测试之前，是否如此还很难说。所以，最好还是提前做好准备，

以防在执行任务的过程中某些测试者很快离开。

创建纸质原型

根据已经确定的测试任务，很容易决定向测试者展示哪些界面（在决定产品原型中应该包括哪些界面时要兼收并蓄，因为人们不总是选择最明显的步骤去完成你所设定的目标）。

如果已有的草图并不十分粗糙或太过简单，也可以用它们来创建纸质原型。太过简单的草图让那些对产品不熟悉的人难以看懂，如果用它们创建纸质原型就会引发一些问题。

为了让草绘图和实体模型更适用于可用性测试，你需要为用户界面加上一些产品运行环境的界面元素。例如，如果你正在测试一个网站，那么就为产品原型添加一些浏览器的界面元素（前进和后退按钮、地址栏、标题栏等）。同样，如果你正在测试一个应用程序，就为产品原型添加菜单栏，如果该程序是在桌面操作系统下运行的，你甚至还要添加一些桌面元素。

在创建好屏幕显示内容之后，你就需要考虑各个屏内所显示内容的状态变化了。

如果有人点击了一个下拉菜单会发生什么？你要预先准备好所有需要用到的弹出窗口。有些人喜欢用便利贴来代表弹出窗口，一旦用户启动了某个弹出窗口，他们就可以很容易地把便利贴贴在纸质原型上。

如果用户要在某个区域内输入数据，我们该怎么做？处理这种情况有几种方法，如果你不打算重复利用纸质原型，那么可以给测试者一支铅笔和一个橡皮，让他们直接在纸质原型上填写数据。如果你要重复利用纸质原型（也就是说要进行多次测试），可以使用透明纸来解决这一问题，把透明纸放在产品原型上，让用户在上面进行修改。

最后，用纸张把每一项任务单独打印出来。由于我们不知道每个用户将完成多少项任务，因此要为每个任务分别准备单独的纸张，这样就可以在测试中随时分配任务，或者在需要的时候改变任务的测试顺序。

创建纸质原型的清单

- ❑ 准备用户将在测试任务中遇到的所有屏幕显示内容，不能太过粗糙。
- ❑ 在需要的地方添加一些产品运行环境的用户界面元素（例如，为网站界面设计草图加上浏览器窗口）。
- ❑ 创建测试所需的弹出元素。
- ❑ 为测试者准备一些在产品原型上修改数据的方法。
- ❑ 在独立的纸张上打印出测试任务。
- ❑ 对每个任务进行测试运行，确保你已经准备好了所有可能用到的屏幕显示内容和弹出元素。

测试的准备工作

用纸质原型进行可用性测试，你至少还需要一个人——执行测试任务的人。有时把这个人称为“测试主体”（test subject），但因为并不是对他进行测试，所以我通常称之为“测试者”（tester）。总之，他是对你的设计方案进行测试的人。

我不想在本书中赘述如何招募测试者，关于这一主题的参考资料①已经有很多了。但一般来说，你应该找自己公司之外的人来进行测试。最好不要请自己公司内部的人，他们对你的产品和公司内部使用的术语太过熟悉了，这可能会掩盖产品的可用性问题。尽量邀请产品的目标用户来进行测试，但一般来讲，测试者是不是目标用户并无大碍。基本上公司之外的任何人都可以，包括你的朋友和家人。

应该对每个纸质原型版本进行多少次测试呢？创建纸质原型有点费时间，因此每个产品原型测试只要不少于两个人就行。另外，有了纸质原型之后，在测试中对其进行小范围的调整还是很容易的。最好的做法是邀请三四个人，每个人分别进行两三小时的测试。这样，你就有足够的时间在测试中仔细检查已发现的问题，如果可能，你还要相应地对纸质原型做出一些改变。

尽管你可以独力执行纸质原型测试，不过最好还是有其他人来协助你。因为你创建了纸质原型，也知道应用程序的故事板是什么样的，因此，你是测试中扮演“电脑”角色的最佳人选。这意味着你要在测试中切换屏幕显示内容，给出弹出元素，模拟真实的用户界面。这样，你基本上就没有时间与测试者交流了。虽然在不得已的情况下可以凭一己之力完成所有这些工作，但另外找个帮手是非常有必要的。我们把这个人称为“协助者”（facilitator）。图 11-1 所示是纸质原型测试的典型人员配置。



图 11-1　纸质原型测试中的角色

协助者的任务是向测试者介绍测试流程，给出所要执行的任务，回答测试者可能提出的问题，以此引导可用性测试的开展。

① 尼尔森 · 诺曼的报告就是很好的参考资源，参见 http://www.nngroup.com/reports/tips/recruiting/。

最早的电脑

把人称为“电脑”可能有点滑稽可笑，但在历史上，“电脑”一词曾用来表示某一类人。“computer”（计算者）一词最早在 1600 年左右被提出来，用来形容那些计算数字的人。

直到第二次世界大战期间，才开始有机器来完成这项任务。那时，克兰德·楚泽和阿兰·图灵等人创建出了第一台计算机器。随着这些机器逐渐流行开来，“电脑”一词的含义才有所改变。

电脑	控制纸质原型，对用户的输入做出反应
协助者	向测试者介绍测试流程，给出任务，回答测试者提出的问题
测试者	根据给出的任务与纸质原型进行交互

协助者应该在测试过程中记录笔记（如果你对整个测试过程录了视频的话），并把录像的时间节点添加到笔记上，这样笔记中的每个事件都可以与录像中发生的事情相匹配。测试过程中做好笔记有很大的帮助，这样你就不必再浏览数小时的测试录像来寻找有用信息了。

对整个过程进行录像是个不错的主意，这样可以在测试完成后再返回去浏览整个测试过程。回顾测试流程，能让你具体地关注测试者在测试过程中的行为，让你洞察他们遇到的各类问题。通常，这些都是你在测试中不会注意到的小细节。另外，如果你不是相关问题的决策者，如果录像说明测试者在同一任务中屡次失败，你就可以借此来说服那些不相信用户界面存在问题的人。

在本书中，我一般不会涉及与法律相关的内容，但是，如果要录像的话，一定要让测试者填写一份同意录像的表格[①]，协助者也要征得测试者的明确同意才能记录笔记。

测试准备清单

- ❑ 招募三至五位测试者，分别为每个人规划两个小时的测试任务
- ❑ 寻找并训练协助者
- ❑ 如果你要自己记录，准备好记录方法
- ❑ 为测试者准备一份同意录像的表格

准备测试者

让测试者坐在扮演“电脑”的人的对面，这样，“电脑”就可以把产品原型的变化呈现给测试者。让协助者坐在测试者的旁边稍微靠后一些，这样才不会挡住测试者的视线。

一旦所有人就绪，协助者要做的第一件事就是向测试者解释：“我们要测试的不是你本人，

① 如果你没有此类表格，我建议你找一个熟悉当地法律条文的律师帮助你撰写一份。但要确保测试者能够理解表格中的所有法律规定。

而是用户界面。”这是所有可用性测试中都要介绍的基本内容。关于更多信息，你可以参考第 28 章。以下是一个简短的开头介绍：

> 嗨！你好，我是软件设计师迈克。这是我的朋友桑德拉，也在设计部门工作，她将协助我开展这次测试。
>
> 首先，感谢你花时间协助我们进行这次测试。
>
> 今天，我们要对本公司产品的新设计方案进行测试，看看它是否像我们设想得那样有效。需要声明的是，我们要测试的对象是设计方案，不是你本人。这个新方案还没有在设计团队之外使用过，因此我们希望通过观察用户与设计方案的交互过程发现一些问题。如果你在测试中遇到困难，或是某些东西出了问题，不要担心，这正是我们希望出现的情况。
>
> 如你所见，我还没有开始实施这个方案，到目前为止它还停留在初创阶段。我们想在开始编写代码之前消除所有问题。今天，我的朋友桑德拉将扮演一台电脑——她将做所有电脑应该做的事情。由于扮演的是电脑，所以她在此过程中不能说太多话，但她会像一台真的电脑一样，根据你输入的内容对产品原型做出改变。在你与我们的设计方案进行交互的过程中，请告诉我你大脑中的所有想法和疑问。因为我们想看看当我们不在场的时候，用户是如何与产品进行交互的，所以我通常不会立即回答你提出的问题，但这样做一样有助于我们了解你大脑中正在想的事情。
>
> 如果你允许，我们将记录这次测试过程。这样，我们可以回顾整个测试过程，找到改进设计方案的方法。我们保证不会以任何形式公开录像内容。
>
> 对于整个测试过程，你还有其他问题吗？

然后让测试者签署同意录像的表格。

对于要介绍的内容，你最好事先做好笔记或列出一个清单，确保协助者介绍了所有相关信息。因为要介绍的东西实在太多，协助者很容易忘掉某些内容。

接下来，协助者立即向测试者介绍纸质原型，解释测试者所看到的东西，比如“这是我们网站的主页。”然后介绍如何与之交互：

> 你可以像操作电脑一样对纸质原型进行操作。可以用手指点击，也可以像用鼠标一样进行拖拽。如果你要输入某些信息，直接用铅笔在原型上写就可以了，不要担心，我们还有另外的拷贝。如果要删除某些东西，就用橡皮擦把它擦掉。如果要更改一些内容，我们的“电脑”将会完成这项工作，比如替换掉屏幕显示内容或加入一些弹出内容、菜单等。我再声明一次，不要感到拘束，你可以大声说话。但“电脑”只会对点击和输入有所反应。

在这个过程中，你可以用草图向测试者举例说明什么是点击、拖拽、输入和擦除。

协助者需要说明，测试者只能和他进行交流，不能和“电脑”说话。如果测试者在测试过程

中试图和电脑交谈，确保只有协助者才能对此做出反应。再次强调，你一定要指出测试的目标是设计方案，而不是测试者。

观察者

到目前为止，我提到的三个角色分别是电脑（控制纸质原型的人）、测试者（与纸质原型进行交互的人）和协助者（引导测试的人）。对很多小公司而言，执行纸质原型可用性测试通常有这三个人就够了。但在较大的公司中，让更多的人参与观察整个测试过程是很有价值的。就我的经验而言，程序员和管理者在观察用户与产品进行交互的过程中都有很大收获。

我们把观察测试的人称为“观察者”（observer）。

对于纸质原型测试而言，观察者可以与其他三个角色坐在同一个房间内，也可以坐在另外的房间，或者在测试完观看视频录像。如果在同一个房间观察，观察者一定要知道，不能以任何方式打扰或影响测试者。我一般认为把观察者和测试者放在同一个房间里是不好的做法。但纸质原型的测试者需要一个“电脑”和协助者，再多两三个观察者在旁边或许不会有什么影响。

告诉观察者在测试过程中要记笔记。在测试完成之后，你要花费一个小时左右浏览他们所记录的问题。观察测试的人越多，能发现的问题就越多。你可以用姻亲表（affinity diagram）对多个观察者发现的问题进行优先度排序。卡洛琳·施耐德的优秀著作《纸质原型：快捷简便地定义和完善用户界面》（*Paper Prototyping: The Fast and Easy Way to Define and Refine User Interfaces*）介绍了与此相关的内容。

执行测试

首先要介绍第一项任务：

> OK，介绍完了这些，现在我们就开始对设计方案进行测试。我在这张纸上写下了一项任务，你先花点时间读一下。如果准备好了，就可以与我们的设计方案进行交互。

在测试过程中，协助者一定要谨记以下几点。首先，不要影响测试者。这一点很重要，特别是在测试者遇到困难或开始提问的时候。如果很明显测试者无法自己解决问题，给予提示（或者直接执行下一项任务）是可以的，但一般来说，协助者一定不能引导测试者。

大多数时候，协助者可以向测试者提问。再次强调，一定要避免不经意地影响测试者，不要暗示测试者如何使用产品原型。产品原型中一定要杜绝出现术语，提问测试者的用词越常规越好。比如：“你在想什么呢？”如果测试者被难住了，可以提问：“告诉我你在屏幕上看到了什么？”更多可用性测试中常见的错误，请阅读第 31 章。

通常来说，只要测试者在全身心地关注界面，最好保持沉默。毕竟，人们在家里使用你的产

品时，总没有人在后面看着，不时地询问他们使用产品的感受吧。

协助者要避免做出任何让测试者感到不安的举动。有人看到他的错误已经让他很紧张了，不要再加剧这种感受。如果协助者看到测试者很沮丧，可以打断一下，给予一点帮助、鼓励，或是让他休息片刻。如果测试者向你求助，或开始因为问题而自责，或者被难住，说明你有必要干预一下。在给予鼓励或是允许休息时，一定不能表现得像是施舍恩惠一样。

“电脑”的任务是根据测试者的输入更新产品原型。如果一切按照计划进行，纸质原型测试主要包括：用新的显示内容替换掉当前屏幕显示内容，根据需要增加或移除界面上的便利贴等。在某些情况下（例如，当测试者使用“搜索”功能时），“电脑”还要擦除某些内容或是在界面上写下某些东西。对于数据量很大的应用程序，特别是具有出色的“搜索”功能的软件，就有必要为测试者可能会选择的路径准备一些已经填充好数据的界面。

有时，测试者可能会走到任何人都预料不到的路线上去。此时，“电脑”要迅速创建一些屏幕显示内容的实体原型。如果测试者偏离了预先设定的测试路线，协助者就要介入进来，阻止测试者继续进行，或将其拉回到预先设定的测试路线上。

在测试者完成第一项任务之后，协助者和“电脑”可以花几分钟询问那些不想（或不能）在测试中问的问题。例如：“你在点击‘购买’按钮之前犹豫不决，能记起来当时看到了些什么吗？”我个人建议协助者不要在测试中问太多问题，而是等到测试完成之后再提问。这样，协助者就基本不会影响到测试者的行为。但这样做的缺陷是测试者通常无法清晰地回忆起他在测试中做了些什么，以及为什么这样做。

第一项任务完成之后，为测试者分配第二项任务，让他继续进行测试。

当最后一项任务完成，或者测试前规划的时间即将用尽时，征求测试者对于产品的观点（如果你想问的话），然后结束测试。他们的观点一般不是我们想要的东西，但测试者有时会在测试完成之后冒出一些非常有趣的想法。例如，你可以问测试者不喜欢设计方案的哪些方面，他以后是否会使用这款产品，用产品来干什么。

询问测试者对整个测试过程的感受是个好主意。作为一种元测试，测试者指出的问题通常是改进未来可用性测试的有用信息。

最后，再次感谢测试者花费时间参与测试，让他知道此次测试结果对你有很大的帮助。

分析测试结果

不要对测试结果做任何统计学或形式化的评估。可用性测试提供的是定性数据，不是定量数据，对这类测试的结果进行形式化的分析会产生误导作用，也是毫无必要的。实际上，即便只是粗略地浏览一下可用性测试的视频录像，通常也能明显看出问题所在。

何时停止测试？

你如何知道什么时候自己的设计才能足够好，不用再进行可用性测试了呢？

实际上，测试是不会停止的。它是开发过程中一个持续不断的部分。在开发过程中，利用纸质原型测试剔除问题之后就要开始编写代码，那么你就要从纸质原型测试逐渐进入到代码运行测试。但是每次做出重大设计变更时，纸质原型测试仍然是有效的工具。如果选定了一个可用的、通过了测试的设计方案，那么先在纸上测试这些变更，再用代码实现，会更加容易。

提示

- 如果想知道如何改进草图和实体模型，就要找真人进行测试。
- 测试没有必要很复杂。向参与测试的人展示草图，然后问一些简单的问题。
- 在更广泛的测试中，针对产品的关键部分定义测试任务。至少准备五六项任务，每一项任务的执行时间为二至十分钟。不能指定任务的完成方式。
- 把各项任务分别打印出来。
- 招募三到五个测试者，分别为每个人规划两小时的测试任务。
- 你可以自己执行整个测试，但最好还是由你来模拟“电脑”，再找一个人作为协助者。
- 准备测试者在完成任务的过程中可能用到的所有屏幕显示内容，这些内容不能太粗糙，确保不熟悉产品的人也能读得懂。
- 在需要的时候，添加一些运行环境的用户界面元素（比如网站草图周围的浏览器窗口）。制作可能会用到的弹出元素。
- 对每一项任务进行测试，确保已经准备好了所有可能用到的屏幕显示内容和弹出元素。
- 在测试开始时，向测试者解释整个测试流程。强调你所测试的是设计方案，不是测试者本人。
- 将所有要介绍的内容列一个清单，在介绍的过程中核对这些内容。不要忘了让测试者签署同意录像的表格。
- 向测试者解释如何与“电脑”交互。允许测试者在原型上写写画画。
- 在测试过程中，不要影响测试者，不要让测试者感到不安。在测试者变得沮丧时介入其中。
- 完成测试后，做个简短的总结，感谢测试者的帮助。

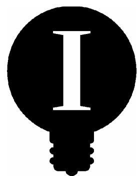
延伸阅读

卡洛琳·施耐德曾写过一本关于纸质原型的权威著作——《纸质原型：快捷简便地定义和完善用户界面》。如果真的对纸质原型感兴趣，你应该读一下这本书。

Userfocus 网站上有一篇关于纸质原型的简明的文章，其中列出了很多资源的链接地址[1]。

11

① 参见 http://www.userfocus.co.uk/articles/paperprototyping.html。



第 12 章 写实主义

随便拿起一台现代一点的设备，比如平板电脑或智能手机，你很快就会发现，设计者做的大量工作都是为了使这些产品的用户界面更具真实感。

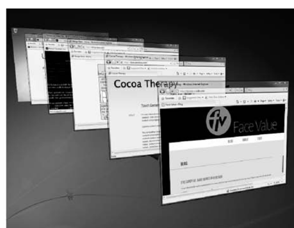
你能看到阴影、渐变、3D 效果和材质。屏幕上显示的元素和整个应用程序都基于实物，甚至连用户交互也都仿照现实：你可以触摸并移动滑块，切换开关，如果推一下某个可滚动区域，它就会持续滚动一会，滚动速度逐渐变慢，就好像真的有摩擦力存在一样。平板电脑上看书软件的界面与真实的书十分相似，日历软件的界面看上去也和真实的纸质日历别无二致。



Palm Pre手机界面使用了阴影、材质和透明效果



苹果iPad上的日历应用以真实的日历为原型，包括撕去了部分纸张的效果



微软在Windows 7中采用了阴影、材质和透明效果，甚至还为操作系统的某些功能使用了3D动画

与现实世界不同的是，应用程序和网站不受自然法则的束缚，它们无所不能。毕竟，它们只是屏幕上一堆闪闪发光的点而已。如果某个人搬起一块石头，然后松开手，他肯定知道会发生什么事情。但如果他触摸了屏幕上显示的一个像素点，程序员为该像素点设定了什么样的行为是无法预料的。正因为如此，赋予应用程序或网站一种实物的外观和行为是一种非常好的做法。这种一致性有助于人们理解产品的工作原理，让人们知道现实世界的法则同样适用于手中的产品，还能帮助人们理解用户界面所能提供的交互方式。

写实风格的设计细节能向人们传达产品支持的操作方式（也称“功能可见性”，参见 9.5 节）。产品越接近于现实世界，用户就越容易搞清楚其工作方式，也越容易形成使用产品的正确心理模型。

如果你对写实主义不屑一顾，那么设计出的用户界面只会让用户迷惑不解。

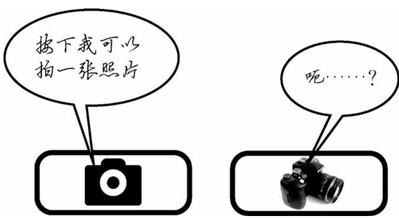
12.1　符号

当前用户界面中的大部分视觉元素都代表某些行为或概念。用户界面中的小铅笔并不代表现实世界中的铅笔，而是表示“编辑”这一概念。“停止”标志也与现实世界中让汽车停止的标志有不同的含义，它用来向你发出警告，提醒可能会有错误发生。

你认为这个图标想要传达什么样的意思？没有任何语境的话，用户根本不能确定其含义，但最有可能的解释是，激活这个按钮可以拍照片。

这个按钮呢？很明显，把拍照按钮设计得更像现实中的照相机并不能让用户更容易地理解其含义。实际上，用户在考虑这个按钮的含义时，反而会因为细节过多而困惑。

漫画家斯科特·迈克劳德在他的著作《理解漫画》（*Understanding Comics*）中指出，为图像加入细节会减少其普适性。带有现实中的照相机形态的按钮不“代表”任何事情，它只描绘出了一个具体的照相机而已。与此相反，缺乏细节的按钮设计只是一个原型，它并不描绘一个具体的照相机，而是传达出“照相”这一概念。



然而，细节太少会让用户更加难以理解你要表达的准确含义。



12

关于符号

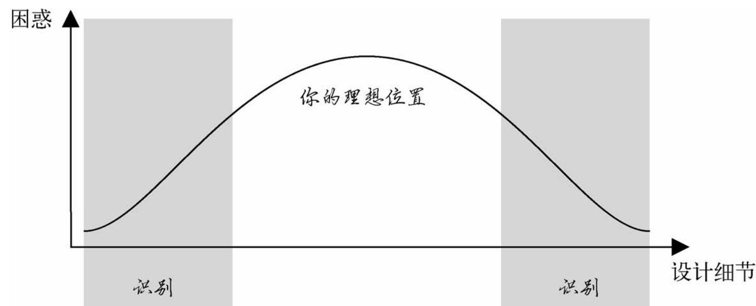
如果符号这么复杂，为什么我们还要用它？去掉符号和图像，直接用文字标签岂不是更方便？

科林·威尔在其著作《信息可视化》（*Information Visualization*）中对人类是否真正能迅速识别图片进行了研究，他指出：

> 在较短的时间内，视觉图像更容易识别。这一事实表明，用户界面中的图标更容易让人记住，能帮助我们回忆起复杂系统的某些功能。易于识别的图标能激活人类大脑长期记忆语义网络中的相关概念。

换句话说，图标让产品更易于使用。

与往常一样，你可以尝试寻找以上两种极端状况间的黄金分割点。在以下这张完全不科学的图表中，某个中间位置是你的绝佳选择：



可用性测试能帮助你找到这个黄金分割点。

12.2　实物的虚拟版本

尽管目前很多用户界面中的视觉元素都用来表示动作、任务或是概念，但也有些视觉元素用来表示实物的再现（representation）。与铅笔图标实际上表示“编辑”按钮不同，音乐软件中的旋钮图标并不表示现实中的旋钮概念，而是表示一种与现实世界中的旋钮工作原理相同、功能相同的旋钮元素。同样，一些电子书软件也使用现实世界中打开的书作为用户界面。这些软件中的书并不代表“阅读”这一概念，而是用来实现与真书一样的功能（如图 12-1 所示）。

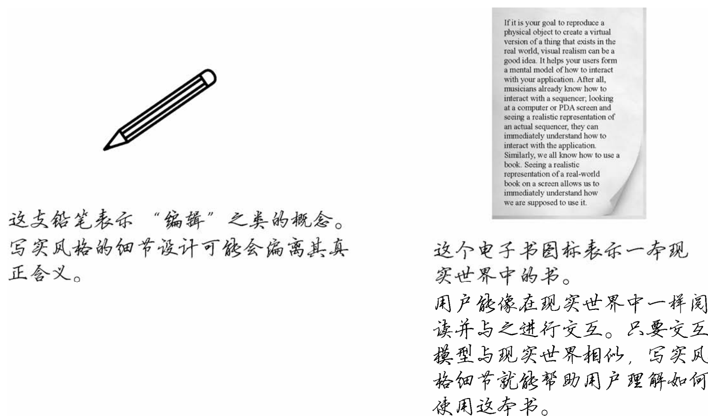


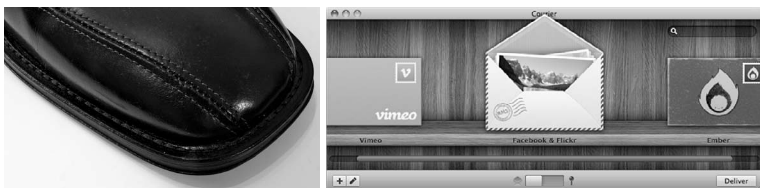
图 12-1　图标设计应该表示概念还是真实的物体

视觉写实（Visual realism）——复制现实世界中的某个物体，创建其虚拟版本——是个非常不错的做法。它能帮助用户形成与产品进行交互的心理模型。毕竟，音乐家已经掌握了使用合成器[①]的方法，只要看到电脑或 PDA 屏幕上显示的模拟现实世界中合成器的软件，立刻就能明白如何与用户界面进行交互。

① 图中合成器是 Korg 的 iELECTRIBE 软件，更多信息，参见 http://www.korg.com/ielectribe。

Skeuomorph

在设计师谈论写实风格的用户界面时，他们有时会将这种界面称为 skeuomorph——只保留原始物体纯粹装饰性细节的新物体。例如，某些鞋的组成部分是粘在一起的，但仍然用线缝了一下，这些线没有实际功能，纯粹为了装饰。类似的例子还有电子邮件应用程序里带有邮资的邮票（下边右图是 Mac 电脑系统上的图片分享软件，名为 Courier）*。



图中的缝纫线和邮票都不具备实际功能。

* 更多信息，参见 http://www.realmacsoftware.com/courier。

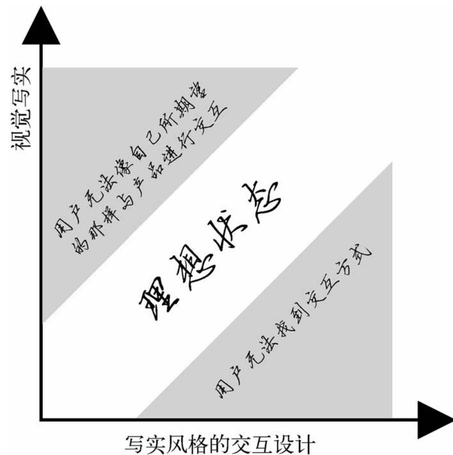


然而，让用户迅速形成固定的心理模型并不总是件好事。这会让用户在使用产品之前就对产品的工作方式产生某种特定的预期。也就是说，如果你采用了写实主义的视觉元素设计，那么交互方式的设计也应该写实。

在设计一款阅读软件时，如果用现实世界中书籍的特征作为用户界面，只在其中加入“前进”、“后退”按钮，让用户从一页转到另一页，这样做是完全不够的。如果用户在屏幕上看一本逼真的书，他们会期望能够用手指快速翻过书的纸张，还期望能够通过查看两边纸张堆积的厚度来确定自己读到什么位置了，因为这正是现实世界中书的使用方法。



相反，在非写实主义的用户界面中采用写实风格的交互方式也未尝不可，但是人们可能无法找出产品的交互方式，除非你引入了写实风格的元素（比如材质）来给予提示。这种用户界面缺乏充足的功能可见性。



如果你正在创建的应用程序或网站再现了现实世界中的物体，那么采用写实主义的视觉设计和交互方式就可能做出出色的设计。不过，这种方法也有一些潜在的问题。

12.3 模拟自然约束

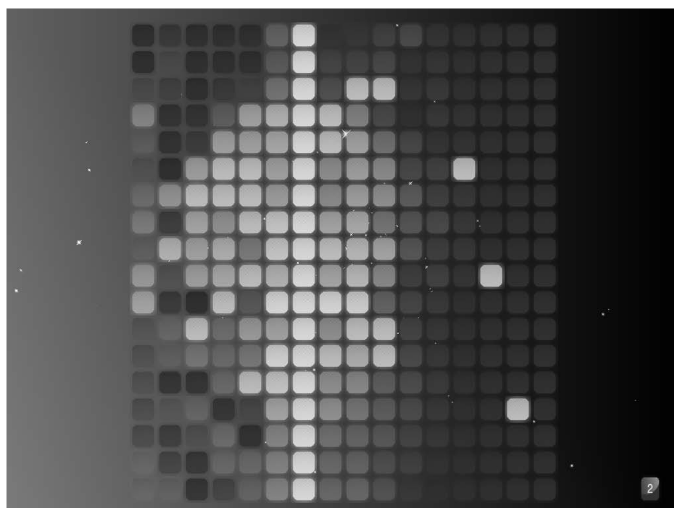
虽然写实风格的用户界面元素能让用户迅速形成正确的心理模型，但这些元素也带来了一些束缚。Instapaper 软件的开发者马克·阿门特（Marco Arment）在他的文章《过度使用界面隐喻》（*Overdoing the interface metaphor*）[1]中指出了这一问题：

[1] 关于该文章，参见 http://www.marco.org/441168915。

> 找到现实世界的复制品和产品可用性之间的平衡点非常重要。实物通常有其特定的运作方式，并有其道理，我们应该保留这些方式。但大多数时候，实物以其特定方式运行是因为受到物理、技术、经济或实际用途等方面的限制，没有更多选择。

可用性咨询师布鲁斯·唐格纳兹尼（Bruce Tognazzini）也有类似看法，他说："在业余爱好者手中，依葫芦画瓢式地模拟现实世界中飞行器的操作方式通常会让他们更远地偏离正确的操作方式。"[①]

前面的 Korg 合成器看上去和实际的合成器一模一样。以下是另一款合成器——iPad 上的 Beatwave[②]的界面截图。



它看上去和真实的合成器毫无相似性可言，其交互方式也与现实世界中合成器的使用方式相差甚远。

下面是另外一个例子。现实中的日历载体是几页有限的二维纸张，由于制造商不知道某个特定用户如何利用时间，因此每一天都得设定成相同大小的页面，并且包含的时间长度要超出用户所需，或者把时间全部留空，由用户自己写入数字。如果某个用户必须在多个地方记录每一个变化，那就很难保证多个视图的一致性。因此，现实中的日历通常都不支持以不同的方式浏览同一数据。

电子设备上的日历就不会受到这些因素的约束，你可以以任何方式浏览日历。例如，以多种不同的视图浏览同一数据，包括以混合缩放比例的方式浏览，既能看到今天的详细日程，又能看清楚本周其他时间的安排。

① 关于该文章，参见 http://www.asktog.com/readerMail/1999-06ReaderMail.html。

② 更多关于 Beatwave 的信息，参见 http://collect3.com.au/beatwave。



在很多情况下，不要试图在电脑屏幕上模拟现实世界的物体及其约束条件，而是最好找一个利用电脑技术优势创建的用户界面。Windows Phone 设计团队在为 Windows Phone 7 创建 Metro 设计语言时就在这方面做得很好。该团队的目标是创建出“真正的数字化产品”。Windows Phone 设计团队的成员迈克·克鲁泽尼斯基（Mike Kruzeniski）写道：“某些 UI 应用了 skeumorphic 的阴影和光泽效果，试图模拟真实世界的物质和对象。但用户界面是用像素点创建而成的，因此，在 Metro 中，我们尝试完全抛弃 skeumorphic 的这些效果。”[①]

这听上去有点像假二歧分类法（false dichotomy）[②]。通常，最好是既使用来自于真实世界的设计素材，又充分发挥数字设备所能提供的自由度。以 Palm Pre 的日历应用来说，它的布局有点像真实的日历，但也具有一些现实世界中的日历无法实现的功能。例如，它把空闲时间压缩在一起显示，这样，用户就能只看到一天中忙碌的时间。



① 更多信息，参见 http://windowsteamblog.com/windows_phone/b/wpdev/archive/2011/02/16/from-transportation-to-pixels.aspx。
② 生物学术语，将特征不同的生物用一分为二的方法逐步对比排列进行分类。——译者注

提示

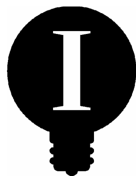
- 为设计方案添加写实细节，这样可以向用户传达可行的操作方式或功能可见性。
- 图标的设计不能太过写实，否则会失去其意义。
- 复制真实物体能帮助用户迅速形成正确的心理模型。
- 基于真实物体的写实风格图形和交互设计应该齐头并进。
- 某些时候，你不必复制真实物体的约束条件，因为屏幕上的物体根本不受自然法则的约束。
- 可以将现实世界的产品不具备的高级功能与有助于用户使用的写实元素相结合，但你要让写实的程度显而易见，这样用户才能形成正确的心理模型。

延伸阅读

斯科特·迈克劳德的《理解漫画》深入剖析了人们如何“阅读”图像。如果你想学习更多的图标设计知识，我向你推荐艾德里·安弗鲁泰格的《标志和符号的设计及含义》（*Signs and Symbols: Their Design and Meaning*）。

麦克斯·斯滕贝亨在其文章《UI设计中的赏心悦目和简单实用》（*Eye Candy vs. Bare-Bones in UI Design*）中谈到了界面设计的吸引性与功能性之间的权衡关系。[1]

① 参见 http://facevalue.virb.com/ blog/text/12959219。



第 13 章

自然用户界面

13

自然用户界面（natural user interface，NUI）是指那些摒弃传统 GUI 界面的设计原则，支持基于自然世界创建交互设计的人机界面。自然用户界面中没有窗口、按钮和菜单，而是基于真实的物体和手势实现交互过程。它不再需要鼠标和键盘，而是依赖于多点触摸屏、摄像头、麦克风、笔和其他支持人类通过手和手指、声音或身体的移动直接与界面进行交互的设备。

CLI 命令行界面 基于预先定义的一系列文本命令的交互	GUI 图形化用户界面 基于隐喻的交互，用图标表示数据和命令	NUI 自然用户界面 基于对真实物体进行直接、自然的操作的交互

命令行界面要求你必须记住严格的语法。图形化用户界面对此进行了改进，呈现出了在特定时间内所能做出的交互，但你必须记住屏幕上显示的每个元素的含义。自然用户界面则建立在你已熟知的事物之上，在理想情况下，你不需要记住任何东西，不需要记住如何做某些事情（与现实生活中完成这些事情的方式一样），不需要记住某个元素代表什么含义（它们与现实世界中相同事物的含义一样）。

试图复制现实世界是一个极佳的目标，因为这让学习和使用产品变得更加容易。但也存在一些问题，我已在与心理模型相关的章节（第 9 章）以及与写实主义相关的章节（第 12 章）中指出了几个问题。本章将阐述一些前面未介绍、与自然用户界面相关的概念。

13.1　避免使用魔法手势

如果选择使用手势实现交互，就一定要确保它能以来自于真实世界、用户能够理解的方式直接、立即作用到屏幕上显示的物体。当用户与系统进行交互时，只有系统立即给予反馈，基于手势的交互方式才会起到非常好的效果。在《设计手势界面》（*Designing Gestural Interfaces*）中，丹・塞弗指出：

> 我们已经习惯了在操纵实际物体时收到即时的反馈。（……）在与手势界面进行交互时，用户想要知道系统是否已经听到并理解了给予它的所有命令，这就需要反馈机制。对于人类直接对手势界面做出的每一个动作，无论多么轻，系统都要给出一些反馈，无论在任何时候，都要尽可能地快。

用户直接操纵物体，而不是发出命令然后观察结果，这种交互方式称为“直接操纵”（direct manipulation）。这一术语首先出现在本·施耐德曼的文章中，他本人是计算机科学领域的教授，就职于马里兰大学的人机交互实验室。[①]

施耐德曼指出，允许人们直接操纵物体使得用户界面更易于学习，也能使用户更加关注自己的实际任务，而不是用户界面。

例如，点击文本内容，然后移动手指滚动文字内容，这样的动作能产生即时的反馈：用户一开始移动手指，文字内容就开始滚动。如果用户做出了错误的操作，他马上就能很清楚地看到自己的动作是无效的，从而尝试另外的操作方式。[②]

但是，如果要求用户画出一个“S”型符号来保存文档，这样做是不太好的。正如哈利·波特用它的魔杖发出咒语一样，用户必须在动作生效之前完成整个手势。用户在做出手势的过程中没有收到有用的反馈信息，如果手势失效，用户无法明显地看出为什么系统不能识别手势。

这类“具有魔法的手势”强迫用户记住手势与命令的对应关系，记住如何做出有效的手势。这样的话，它和命令行界面一样糟糕，只不过用户是通过在屏幕上移动手指，而不是通过键盘输入命令。



与此类似的是，自然用户界面常常避免使用“传统”的、经证明有效的界面元素，比如可见菜单和按钮。产品的功能有时被隐藏起来，用户很难发现其访问方式。例如，触摸并持续按住用户界面一段时间，才会有菜单弹出。

问题在于，直接操纵、可见性和简单性这三者对于自然用户界面比对于其他形式的用户界面更为重要。用户需要在不求助于使用手册的前提下了解该如何与屏幕上的事物进行交互。

① 更多内容，参见本·施耐德曼的论文 Direct Manipulation for Comprehensible, Predictable and Controllable User Interfaces。

② 虽然这种交互方式不那么容易被用户发现，不过可以通过设计，让屏幕只显示某个列表最后可见内容的一部分，这样就能提示用户，后面还有更多内容。

13.2　手势的识别

在观察人们使用带有触摸屏的设备时，你迟早会发现他们在执行“滑动”这一最简单的手势时屡屡遭遇失败。很多应用程序都有这个动作，尤其是大部分读书软件，都用这一动作从书籍的当前页进入到下一页。如果这一动作真的如此简单，在大部分应用程序中都得到了应用，为什么人们经常无法正常做出这一手势呢？

原因之一在于，不同的应用程序对这一手势的识别略有不同。同样的滑动手势，在某一程序中能够识别，而在另一程序中却不能识别。因此，人们可能已经习惯了某个程序中的滑动操作，如果这一操作在另外的程序中无效，他们就会很迷惑。

另一个更为重要的原因在于，通常没有有用的反馈信息来告知用户设备何时“接受”了操作手势。用户不知道该在何时停止操作手势，因为他们不知道在何处放开手指会取消操作，不知道滑动到何处才能让程序跳到下一屏。当然，不止滑动操作存在这些问题，所有的操作手势都存在着类似的问题。唐纳德·诺曼对这些问题的描述为：[①]

> 并没有重要线索表明手势是人机交互设计成功的必要条件。因为操作手势的延续时间非常短暂，也没有任何路径记录。这意味着，如果某人执行了一个操作手势，无论是设备没有反应还是出现了错误反应，几乎都没有有效的信息能帮助用户理解为什么出现了这样的状况。

此类问题的解决方法有时是不明确的，但通常都归结为要让用户知道电脑何时已经识别了操作行为。

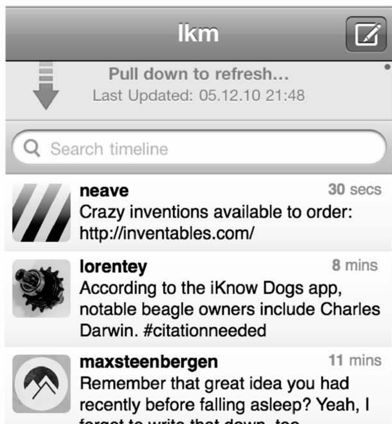
简单的事物，如开关就可以采用这种方法。在配有鼠标的电脑上开启一个开关非常容易，用户只要点击一下就行了，几乎不会出错。



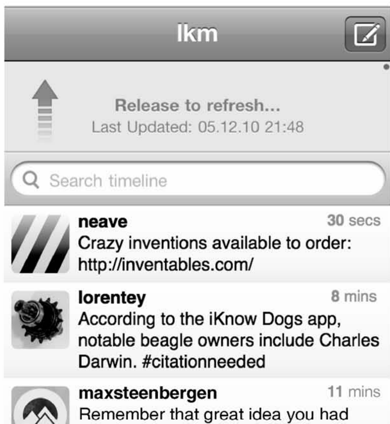
在自然用户界面中，交互的过程是逐渐进行的，其中的变化极其微妙。手势没有明显的开端和结尾，因此用户无法明确知道该在何时停止滑动开关。向用户提供操作手势何时成功完成的反馈信息，就能提示用户已经完成了操作，教会用户如何更高效地使用产品系统。

此处有一个官方 Twitter 应用的例子，它会告诉用户应该在何时停止操作手势（图见下页）。

① 唐纳德·诺曼关于自然用户界面的文章，参见 http://jnd.org/dn.mss/natural_user_interfaces_are_not_natural.html。



如果你拖动列表查看是否有新的Twitter信息，该应用会首先要求你拖动屏幕来刷新信息页面……



……只要你拖动的距离足够远，页面就会刷新信息，并告诉你可以释放操作

创建一个简单的手势用户界面也并非易事，但一些指导原则或许会对你有所帮助。立即给出反馈信息，告诉用户系统何时识别了操作手势；如果还没有识别，为什么。简单、能容错的手势最有效。最好采用直线路径的操作，简短的操作手势优于冗长的操作。如果用户需要在某个特定的区域开始或结束操作，把这一区域设定得更大一些。向用户提供简单的方法来撤销操作，以防电脑没有正确识别某个操作手势。

13.3　偶发性输入

不久之前，我把杂货店购物清单从纸上转到了手机应用程序上。很快，我发现自己每次购物的时候都会少买两三件物品。经过进一步的清查，我发现自己并不是真的忘记要买这些东西，而是不小心把它们从购物清单上移除了。当我一只手拿着手机，另外一只手把东西放进购物车时，常常会无意之中按到手机屏幕，这样也就偶然地把某些东西标记成了“已购买”。

无独有偶，当我和朋友第一次玩 Kinect[①]时，游戏会毫无缘由地暂停。最后我们才发现，游戏机后面的人倚靠在了桌子上，Kinect 把这一动作理解为“暂停游戏”。

当然，人们在使用图形化用户界面时也会犯错。比如点击了错误的按钮，原本要执行点击操作，但实际却执行了拖拽操作。但在大多数情况下，这些错误很容易纠正，因为用户能立刻看到有问题发生，他们只要撤销操作就可以了。

① 微软的 Kinect 是 Xbox 360 电子游戏平台的外设，它能让用户通过肢体的移动来控制游戏。触到屏幕会删除某些东西，倚靠桌子会触发电子游戏的某个操作，对着某个手机讲话会激活临近设备的语音识别系统。这些现象很容易发生，但人们通常很难立刻觉察到这些失误造成的影响。

如果用户没有意识到当前出现了问题，他们就不会撤销自己的操作。如果问题的根源不明显，用户很难避免问题再次发生。

在设计自然用户界面时，重要的一点的是你应该花些时间考虑一下：如何才能避免偶发性的输入；当出现问题时，如何告诉用户问题出在哪里；即便用户没有立刻发现错误，该如何让用户撤销那些无意的操作。

13.4　惯例

假如你想给予用户安全感，让他们自由地探索你的应用程序，就得为用户提供简便的方法来撤销他们的操作。

如果是为桌面平台（如 Windows 或 Mac OS X）下的软件设计用户界面，实现“撤销”操作是很容易的。现在已经有一些惯例和指导原则，它们能告诉你该如何实现这一操作。图 13-1 是 BusyCal[①]的界面截屏。

但如果产品是自然用户界面，实现这一操作要困难很多。目前也没有太多的惯例或规则能提供帮助。[②]



图 13-1　BusyCal 中的撤销操作

你得自己找出解决以上问题的方案，设计出让人感觉到“自然”的交互方式。但在现实生活中，你无法让时光倒流，改变几秒钟之前的行为。那么，如何才能在自然用户界面中表达这一概念呢？

不幸的是，对于如何设计出“自然的”用户界面，目前还没有简单明了、显而易见的解决方案，但以下几点可以作为设计参考。

- 如果可能，复制现实世界中的行为。让电脑以用户能够识别、能够理解的方式立即做出响应。这样，用户就可以把现实世界的知识应用到虚拟系统之中。

① 参见 http://www.busymac.com。

② 假如人们对自然用户界面的工作原理有一些预期，这些预期通常来自于科幻电影，正如克里斯・诺赛尔（Chris Noessel）和内森・谢德洛夫（Nathan Shedroff）在他们的演讲 Learning From SciFi Interfaces 中所描述的那样。关于该演讲，参见 http://vimeo.com/15233780。

- ❑ 参考流行应用程序的做法。有可能你的用户见过这些应用程序，并且也已经熟悉了它们的交互方式。
- ❑ 制作出不同设计想法的交互原型，然后进行可用性测试，看看人们对它们的反应。

在 NUI 变得更为流行之前，我们仍然还要面对用户界面缺乏通用惯例的情况。这让我们的工作举步维艰，但幸运的是，它也让我们的工作更加妙趣横生。

提示

- ❑ 尽可能允许用户直接操纵产品。
- ❑ 对所有的用户操作给出即时、生动的反馈信息。
- ❑ 把那些并不操纵对象，而是激发命令的用户手势仅仅作为快捷方式，而非主要的交互方式。
- ❑ 确保用户不需要学习复杂的手势。
- ❑ 确保用户不需要做出精确的手势，给予用户输入的自由，但要允许他们撤销错误的输入。
- ❑ 运行自然用户界面的硬件特别容易把用户的无意操作理解为信息输入。尽可能阻止偶发性输入；如果发生了偶发性输入，提供应变方案。
- ❑ 道法自然。
- ❑ 参考流行软件的做法。
- ❑ 定期让实际用户对用户界面的交互原型进行测试。

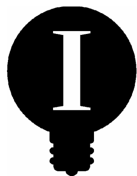
延伸阅读

丹·塞弗的《设计手势界面》中也涵盖了本章的部分内容和很多其他话题。乔希·克拉克在他的杰出著作 *Tapworthy* 中谈到了手势（和很多其他内容）。

弗莱德·毕奇尔给出了一些设计自然用户界面的有效指导原则。[1]

13

① 参见 http://userexperience.evantageconsulting.com/2010/11/ui-guidelines-for-skeuomorphic-multi-touch-interfaces/。

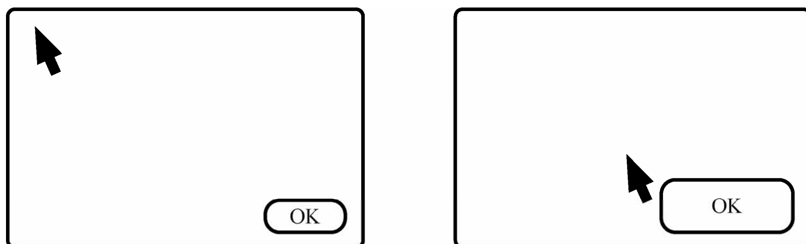


第 14 章

菲茨定律

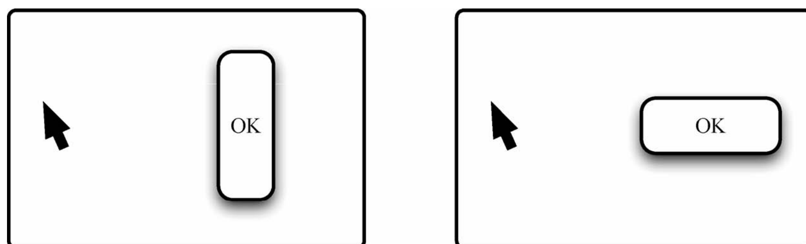
14

在可用性方面，只有少数人们广为接纳的法则。菲茨定律（Fitts's law）就是其中之一，它描述道：（在桌面系统中）用户能快速点击到那些较大或比较接近于鼠标指针的目标。此类目标可能是用户想用鼠标点击的屏幕上显示的图标，或者是用户想要敲击的触摸屏上的按钮。



相比之下，在左边界面中点击"OK"所用的时间比在右边界面中多

动作的方向也是影响因素之一。目标的形状应该与动作方向一致：



同样，在左边界面中点击"OK"所用的时间比在右边界面中多

菲茨其人

菲茨定律是根据它的发现者保罗·菲茨命名的。保罗·菲茨是俄亥俄州立大学的心理学家，后来任职于密歇根大学。

该法则有时写为“Fitts' law”，去掉了所有格的 s，但“Fitts's law”似乎更常见于一些可靠的来源。因此我在本书中使用这种写法，请不要因此向我发邮件抱怨。我向你保证：如果你能改变维基百科文章中的写法，并且使这一变化保持下去，那么我也会改变本书中的写法。

这些观点或许太直接了，但事实上，很多研究结果都表明以上观点是正确的。[1]我们可以用界面中的相关数据计算用户完成某一任务所需的时间，进而计算任务的难度系数（ID，index of difficulty），用它来度量用户完成某一目标的难度。公式如下所示：

$$ID = \log 2\left(\frac{\text{目标距设备指针的距离}}{\text{目标在运动方向上的宽度}} + 1\right)$$

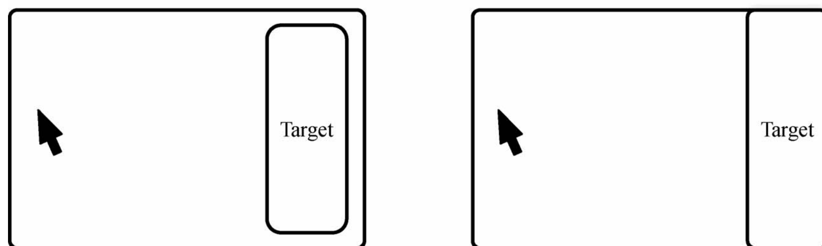
由于难度系数 ID 是通过对数计算的，因此，如果目标初始尺寸较小，很小的变化就能引起任务难度的巨大变化；但对于较大的目标而言，细微的变化产生的影响很小。

虽然该法则本身所表达的含义已经很明显了，但它的一些效果尚未为人所知。让我们来看一些例子。

14.1　屏幕边缘具有无限大的尺寸

如果细想一下，屏幕边缘处的界面元素的尺寸实际上是无限大的，因为无论你朝着屏幕边缘方向把鼠标移动多远，它永远不会超出屏幕边界，总能停留在此处的点击目标上。因此，把重要的界面元素放在屏幕的边缘处具有一定的道理。点击这些元素更加容易，因为人们只要把鼠标甩到屏幕的边缘，它就会自动停在要点击的目标上。

屏幕的角落是特别好的位置，因为此处有两个边界。



在左边界面中点击目标比在右边界面中点击目标需要更多的时间

[1] 关于这方面的研究，请参考 http://www.yorku.ca/mack/RN-Fitts_bib.htm。

不幸的是，在大部分情况下，类似的做法并不适用于触摸屏，触摸屏幕边缘处的界面元素并不比触摸其他地方的元素更容易。用户最容易达到的区域取决于持握设备的方式。



如果你这样握住iPad，容易触及的屏幕区域只有一小部分

当然，在用户拖拽界面元素时，触摸屏的边缘就是一个非常好的目的地了，因为不可能把某个东西拖到屏幕之外的区域。

14.2　放射式环境菜单会减小平均移动距离

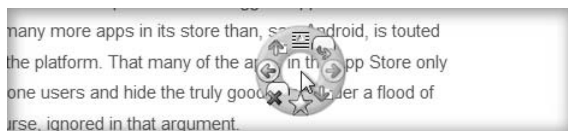
如前所述，屏幕的角落位置很容易点击，因为那里有两条边界。但还有一处更容易点击：指针所处的位置。你完全不必移动指针就可以点中，环境菜单（context menu）就利用了这一点。

如果鼠标指针所在位置处有菜单弹出，在指针周围布局各个菜单可以有效减小到达每个选项的平均距离。实现这种效果的做法之一是使用放射式环境菜单（radial context menu）。

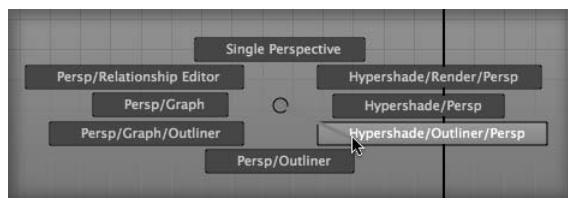
很多游戏都使用了放射式环境菜单。来自于 LucasArts 的《极速天龙》（*Full Throttle*）就是其中之一。



Firefox 浏览器的扩展程序 easyGestures 也使用了放射式环境菜单。



既然放射式环境菜单如此有效，为什么它没有普及开来呢？原因之一在于，我们很难将大量菜单选项集中到一个小圆圈之内。Maya 克服了这一缺陷，它把常规的菜单标签环绕布局在鼠标指针的周围。



有一种解决方案既保留了传统环境菜单的外观，也利用了放射式菜单的优点，那就是拥有多个水平层级的常规环境菜单。我们只要把传统的环境菜单分解成数个更小的命令组，然后利用水平空间把它们布局在鼠标指针周围就可以了。[①]

这类菜单带来的另一个好处在于，它高效地利用了水平空间，你可以同时显示更多的菜单项。这样做以后，你就不必把任何菜单项隐藏在很难用的子菜单中。

圆和直线的感知

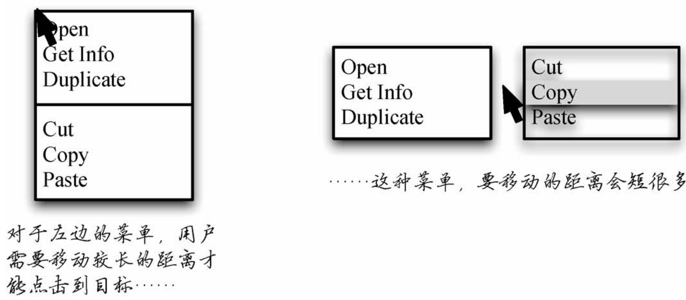
为什么放射式环境菜单没有普及开来？史蒂芬·帕默尔在《视觉科学》中解释了其中一个原因。在说明人类如何感知视觉信息的某个模型的时候，他描述道："视觉皮质第一区域内的细胞拉长了某些感受区域。如果受到某个特定方向或位置的边界或直线的刺激，这些感受区域的反应会非常强烈。"也就是说，人类利用某种特殊的机制来观察直线。此外，我们更擅长于感知水平线和垂直线，其他方向的线条次之。史丹利·科伦在《感觉与知觉》（*Sensation and Perception*）中推测道，由于人类大部分时间都处在直线包围的环境中，经常看到水平线和垂直线，因此我们的大脑特别偏爱这两个方向上的线条。

确实，安德鲁斯在题为 Perception of contour orientation in the central fovea part I: Short lines 的论文中也写道，大部分实验中，人类对接近于水平和垂直方向的感知能力最强。其他研究也赞同这一观点。例如，盖·奥班在题为 Human orientation discrimination tested with long stimuli 的论文中总结道："人类对主子午线附近较窄范围内的方向敏感度好于其他区域。"

长话短说，研究表明：相对于斜线和圆，人类能迅速且正确地感知水平线和垂直线。

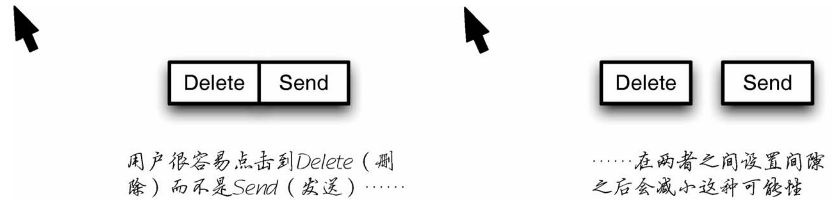
① 这正是我在 Appway 中使用的菜单形式。Appway 是一个商业流程管理系统，参见 http://www.appway.com。

最后，把菜单元素水平布局在指针的左右两侧不仅能减小指针需要移动的距离，还能让用户更容易地点击菜单项。由于指针是向左右移动的，因此指针的移动与目标区域的形状相一致。



14.3　较小目标需要设置外边界

较小的目标难以点击，因此在较小的目标之间设置边界就显得非常重要。否则，用户很可能会错过正确的目标，从而引发错误操作。

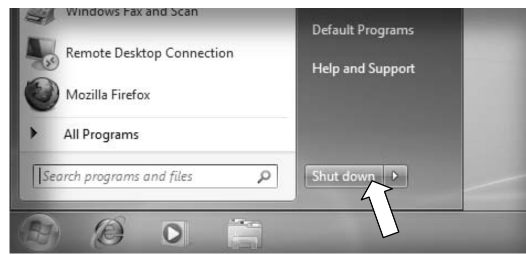


这样的设计同样适用于键盘快捷键。如果破坏性操作的快捷键和非破坏性操作的快捷键是距离很近的两个字母，那么用户很可能会不小心启动破坏性操作的命令。

14.4　有时，界面元素越小越好

把屏幕显示元素设计得大一些会方便用户点击，而把破坏性操作的界面元素设计得小一些能有效降低用户在无意之中点击到的概率。

例如，在 Windows 7 中的开始菜单中，“关机”（shut down）按钮就比其他可点击区域小。



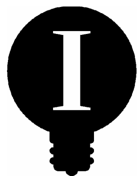
提示

- 如果你想让某些东西很容易点击，就把它们设计得大一些；如果你想让它们不容易被点击到，就把它们设计得小一些。
- 屏幕边缘处的元素更容易点击，屏幕角落处的元素更是如此。
- 指针的运动轨迹应该与所要点击的目标形状一致，也就是说，如果用户水平移动鼠标来点击目标，把目标水平放置会让它更容易被点击到。
- 用户能更快地到达那些接近指针的元素。
- 在不同的可点击元素间保留一些间距。

延伸阅读

对菲茨定律的完美阐述，参见凯文·希尔的文章《菲茨定律可视化》(*Visualizing Fitts's Law*)。[①]

① 参见 http://particletree.com/features/visualizing-fittss-law/。



第 15 章

动　画

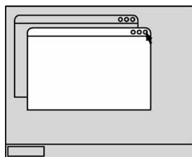
用现在的电脑开展工作绝对是一场视觉盛宴。几乎所有的操作都能引发屏幕上可见的、图形化的变化。当我们写一封信时，会出现图像和字符；当我们在网页上查找信息时，浏览器窗口里的内容会发生变化，呈现我们正在访问的网页；当我们阅读邮件时，信息会打开和关闭。

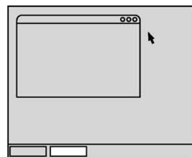
现代用户界面同时包含有非常细微和极其明显的视觉变化。前者比如在我们阅读了一条信息或将其标记为“未读”之后，信息前面会消失或再次出现一个圆点；后者比如我们打开了另外一个网站后，屏幕上的所有内容都会发生变化。人们常常很容易忽略某些变化，或被巨大的变化搞得迷惑不解。

如果使用得当，动画能够帮助用户理解他们的电脑正在发生着什么样的变化。动画还有助于解释屏幕两种不同视觉状态的因果关系，能吸引用户的注意力，让他们注意到那些原本会忽略的东西。动画还能帮助用户形成关于产品工作原理的正确心理模型。

15.1　解释状态的变化

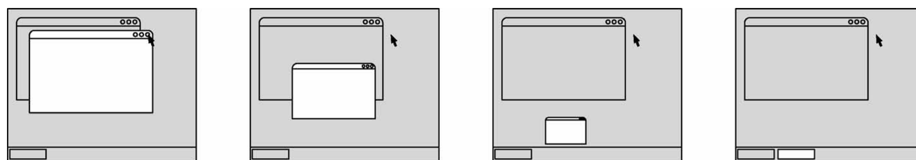
视觉状态的变化很容易让用户找不着方向。如果你正在执行可用性测试，或教某些人使用一款产品，通常你很快会听到“刚才发生了什么？”之类的问题。





我只是点击了这里的一个小小的按钮……呃，窗口跑到哪里去了？

如果听到某人说“我刚才做了什么？”或者“这样做会发生什么？”，或者“我该如何返回去？”，那么有可能是你发现了一个需要用动画来解决的问题。



添加一个简单的动画就能让用户看清楚窗口去哪儿了

模拟两种状态之间变化过程的动画能让状态的改变和两种状态的关系显而易见。

15.2　引导用户的注意力

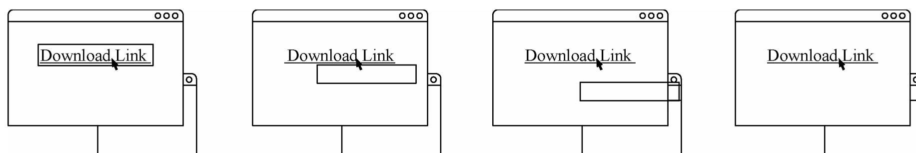
当用户因较大的状态变化而迷惑不解时，他们通常会问自己发生了什么事情。如果他们疏忽了细微的状态变化，典型的反应是："我刚才是否做了什么？它起效了吗？"

你肯定有过以下经历，点击一个链接来下载文件，但什么都没发生，因此你再次点击那个链接，还是什么都没发生。直到你点击了数次之后才注意到已经下载了好几次同一个文件，但电脑却根本没有告诉你这些。



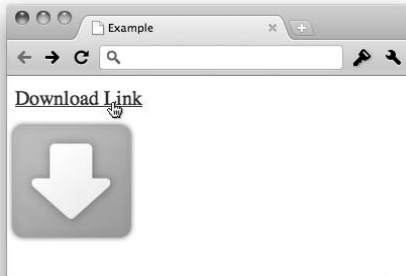
点击了下载链接之后，是否真的发生了什么事情？

对于这类问题，你可以通过动画模拟状态变化来解决，制作动画的目标是夸大状态的改变。当某人启动了下载过程，你最好用夸张的动画把用户的眼球从链接引导到下载窗口中去。



添加简单的动画就能让"点击链接启动下载"这一过程变得很显眼

让用户无法忽略正在发生的事情。例如，Google Chrome 在用户点击下载链接时显示了一个动态的蓝色箭头（图见下页），以此向用户提示已启动一个新的下载，并暗示用户应该在何处查看。



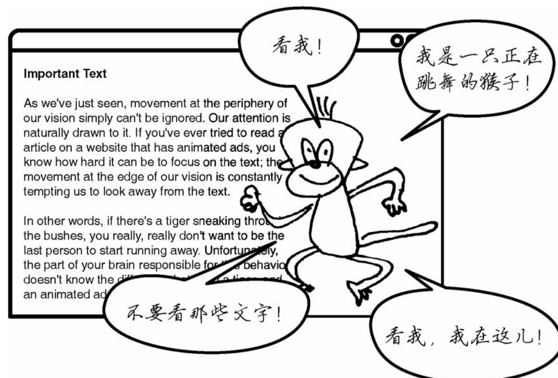
科林·威尔在《信息可视化》一书中指出，吸引用户注意的或许并不是动画本身，而是出现的新目标。他写道：

> 当早期人类男性走出洞穴，有意识地把一块石头打磨成手斧，或女性在草地上采集植被的根茎时，随时监视视野周边出现的事物对于生存来说具有着明显价值。突如其来的移动或许预示着潜在的攻击。当然，进化优势还可以追溯到更久以前。监视视野边缘区域以观察移动的捕猎者或猎物能为大部分动物提供生存优势。因此，最有效的提醒应该是映入视野内，消失，然后每隔一段时间再次出现的事物。

这种行为在我们的进化历史中已是根深蒂固，难以磨灭。

15.3　避免使用不重要的动画

如前所述，视野周边的运动让我们无法忽略，我们的注意力会自然而然地转移到这些运动上去。如果一只老虎在灌木丛中匍匐潜行，你肯定不想是最后一个跑开的人。不幸的是，你的大脑与这一行为相关的部分却区分不开真的老虎和广告动画。

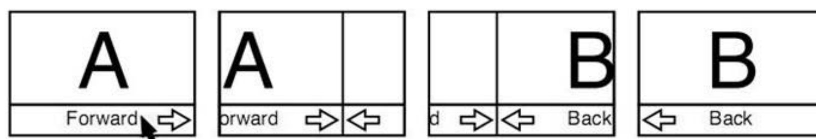


类似上页的动画会让人发疯，让他们无法集中精力于实质的内容。

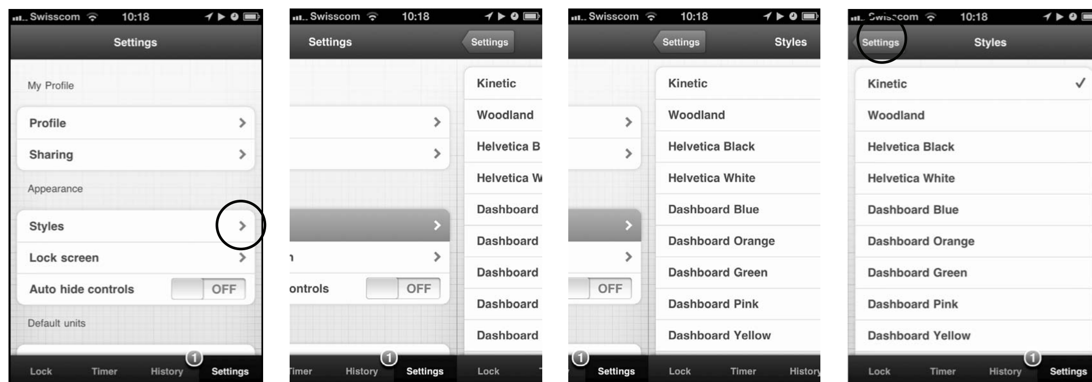
只有当你真正想打断用户，强迫他们关注某些东西，用动画来传递相关信息时，才使用动画，否则就不要用！

15.4 帮助用户形成恰当的心理模型

动画可以用来影响用户对于产品的心理模型。动画所传递的信息应该与用户界面的其他部分保持一致。以通过滑动操作从屏幕右边引入新的内容为例，如果用动画实现这一操作，那么返回按钮的图标应该包含指向左边的箭头。否则，图标就与动画不一致，两者暗示的心理模型相互冲突。



下面是一个来自于 iPhone 的应用 Kinetic[①]的示例。



图标与动画一致。指向右边的箭头把用户带到右边的另一屏内容，反之亦然。

我的 Android 手机的主页按钮就是个动画与实际行为不一致的例子。如果我正在运行一个应用程序，点击主页按钮之后，动画暗示我正在向右边移动，但主页按钮的图标——一个房子——并没有任何方向性的暗示（图见下页）。

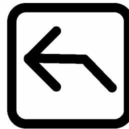
① 该应用可以帮助你记录跑步过程中的相关信息。更多信息，参见 http://wearemothership.com/kinetic。



事实上，如果我点击返回按钮，它的箭头图标暗示我会回到左边的界面，但什么都没发生。



点击主页按钮向右边移动……



……但是点击带有指向左边箭头的按钮，界面并没有向左移动

在新版本的操作系统中，Google 改变了主页按钮的动画。以前人们可能会误认为，通过点击主页按钮退出应用程序后，再按返回按钮就会跳回，这一简单的修复减小了这种误解。

考虑用户的心理模型。使用动画巧妙地强化正确的心理模型，剔除错误的心理模型。

15.5　向卡通漫画学习

卡通漫画已有上百年的发展史。漫画家创造出了一种虽然复杂，但连小孩子都很容易理解的视觉语言。虽然漫画中大量主体和角色都是同时展现，并伴有复杂的动作，但漫画仍然有值得我们学习的地方。张倍伟（音译，Bay-Wei Chang）和大卫·昂加尔（David Ungar）在他们的论文 From Cartoons to the User Interface[①]中指出了卡通漫画值得我们借鉴之处。

实体感

在现实世界中，物体都具有实际形态：大小、惯性、重量和平衡。这些都是人们本来就能理解的特征。在动画中重现这些特征更有利于让用户明白当前正在发生什么事情。

让用户与动画中的物体进行交互，而不要呈现给他们物体的轮廓或其他占位符。例如，当用户在某些程序中拖曳并释放邮件时，这些程序只在鼠标指针下方显示一个单独的图标（而不是被

① 关于该论文，参见 http://research.sun.com/techrep/1995/smli_tr-95-33.pdf。

拖拽的邮件），这样就会破坏电子邮件给人的像纸质信件一样具体实在的印象。





如果在Windows 7中拖动文件，你能看见正在拖动的文件本身

但在Mac中拖动邮件的话，你只能看到被拖动信息的抽象表示

为了让物体具有实体感，动画中不应该有人类能够觉察到的延迟，界面元素的移动一定要流畅。Pixar 的首席创意官约翰·拉塞特（John Lasseter）在他的论文 Principles of traditional animation applied to 3D computer animation 中提道：

> 随着动作速度的提升，位置间的距离会变大。当距离变得足够大时，两动画帧之间的物体就不会再重叠，人眼也开始能感知到独立的图像。

根据经验，如果某个物体在两帧间移动的距离超过了它本身尺寸的一半，那么在运动方向上拉伸物体或在动画中添加动态模糊效果，这时即便物体是在两个位置间跳动，用户仍会把该物体看成实体。①



为了帮助人们把运动的物体看成实体……



……可以在运动方向上将该物体拉伸……



……或在相同的方向上将该物体变得模糊

在现实中，实体从来都不会莫名其妙地蹦出来。张和昂加尔建议，为了保持实体感，物体应该从屏幕外的一个点飞入或逐渐显示在人的视野之内（并以相同的方式从屏幕中消失）。

夸张

你可以利用所有现实中可能的效果，让屏幕上发生的事情变得更加明显。夸大动画的重要部分有时能让动画看上去更加真实。

正如跳远之前需要助跑一样，在动画的起始处，你最好显示一个比动画的主要部分运动幅度小的动作。这是一种特定的夸张形式，称为“预先动作”（anticipatory movement）。

约翰·拉塞特（John Lasseter）曾说过：“没有先兆的动作会显得突然、僵硬而且不自然。”

① 正如凯斯·朗（Keith Lang，Skitch 软件的设计者之一兼首席运营官）所说的那样，动态模糊能让动画更具艺术效果，参见 http://www.uiandus.com/blog/2009/7/2/blur-the-new-black.html。

加速与减速

真实物体不会在动作开始时就具有一定的速度，肯定存在加速阶段。因此，有必要在动画起始处让动作慢一些，然后加速，最后在停止前再让速度逐渐减慢[有时称为缓入（ease-in）、缓出（ease-out）]。这一逻辑同样适用于增长和收缩动画。

同样，现实世界中的物体也很少以直线路径运动。当你扔出一个球，它的运动轨迹是弧形的。对于某些动画而言，模拟这些行为是非常有意义的。例如，你在苹果的应用商店购买一个 Mac 应用程序，它就会以弧形的路径飞入 Dock。

最后，现实世界中诸如弹簧之类的东西会在击中目标后弹回来。在动画中再现这类行为也能加强界面的真实感。

在 2007 年的 MacWorld 大展上，史蒂夫・乔布斯介绍了第一款 iPhone 产品，并作了主旨发言，展示了该设备流畅、富有弹性的滚动界面，他还讲了以下这个小故事[1]：

> 不久之前，我给苹果公司内部的某个人看了这个产品的小样。在此之前他从来没有见到过这款产品。在我演示完产品样机之后，他告诉我说，“你让我满脑子都是滚动。”

关注一下类似的细节，你可以把一个用户勉强能接受、能工作的设备变成一段美好的使用体验。

用户界面并非卡通漫画

记住，我们的目标不是把用户界面变成卡通漫画，而是利用一些卡通漫画使用的工具把界面变得更易于让用户理解，这一点非常重要。

不要因为一些工具和动画看起来很可爱或很闪亮就统统使用。比如 Android 提供的动态家庭场景墙纸，虽然用户使用这些特征可能有其用意所在，但实际上，这些墙纸只会分散用户的注意力，无法显示任何有用的信息或提供任何有效的功能。

有时，为了让产品更有趣而使用动画是可以的，即便这些动画并不能带来实际的益处。但是一定要记住，动画会分散人的注意力。为增加产品的趣味性而牺牲可用性是得不偿失的（更多内容，参见 26.4 节）。

提示

- 用动画把用户的注意力吸引到细微的或不被注意的界面变化上。
- 通过动画让用户理解大幅度的界面状态变化。
- 动画，特别是视野边缘处的动画具有极大的吸引力。不要滥用，它会让人崩溃的。
- 你可以通过动画强化用户正确的产品心理模型。

---

① 参见 http://www.apple.com/quicktime/qtv/mwsf07/。

- 卡通动画有我们可以借鉴的视觉语言：物体应该具有实体感，通过加速和减速营造真实感，有时应该夸大运动效果。

延伸阅读

科林·威尔的《信息可视化》介绍了一些关于动画的信息。张倍伟和大卫·昂加尔在的论文 Animation: From Cartoons to the User Interface[①]也介绍了大量类似的信息。

马库斯·韦伯在他的博客[②]中谈到了在用户界面设计中使用动画的事宜。凯斯·朗撰写了关于在用户界面中使用动画的文章。[③]麦克斯·斯滕贝亨在他的博客[④]中也谈到了这个问题。

最后，约翰·拉塞特的论文 Principles of traditional animation applied to 3D computer animation 里包含有一些关于如何使用动画的极为重要的内容。[⑤]

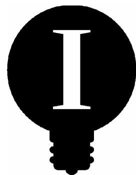
① 参见 http://research.sun.com/techrep/1995/smli_tr-95-33.pdf。

② 参见 http://www.centigrade.de/en/blog/article/animations-in-user-interface-design-essential-nutrient-instead-of-eye-candy/。

③ 参见 http://www.uiandus.com/blog/2009/2/1/interfaces-and-animation.html。他在自己的文章 The Art of Expectations 中指出，人们或许会对某些类型的变化视而不见，参见 http://www.uiandus.com/blog/2008/8/25/the-art-of-expectations.html。

④ 参见 http://facevalue.virb.com/blog/text/12641234。

⑤ 参见 http://portal.acm.org/citation.cfm?id=37407。



第 16 章

一 致 性

16

“一致性”是评论用户界面时使用的流行词，当我们不喜欢某个用户界面设计时，通常会说它“缺乏一致性”。

但是，一致性究竟指什么？

韦氏词典的定义是：“一致性是指产品的部分界面或功能与产品其他界面或产品整体的一致或协调。”简言之，当我们说某个用户界面不一致时，其实是指它与所在系统或该系统中的流行应用程序不一致，或者产品的不同特性之间相互不一致。也就是说，它不同于我们已知的产品，或者产品中不同部分的工作方式各不相同。

16.1 原型的识别

我们讨论的究竟是什么样的不同呢？

通常，我们讨论的都是产品外观的不同。比如，某些东西看上去有点不太对劲，某个应用程序中的按钮与所在系统的默认按钮不太一样，窗口看起来有所差异，滚动条看上去有点奇怪，等等。

视觉上的不一致非常明显，所以人们很容易注意到。但是只关注事物的外在会产生误导。在涉及可用性时，外观并不总是那么重要。人类并不傻，你不必为了让用户能够立即识别就把界面元素设计得完全一样。虽然看上去不太一样，但以下这些按钮都能被用户正确地识别。



（在第 12 章中，我已介绍了用户界面应该详细到何种地步才能容易地被用户识别。）

同类界面元素不必完全一样，只要用户能迅速识别出它们是什么就可以了。

通过简单的可用性测试，就可以很容易地评估出视觉一致性是否重要。你只需要向人们展示

你的用户界面元素（最好在完整的用户界面环境下进行测试），并询问他们这些元素是什么。如果用户能识别出这些元素是什么，你就不必为视觉上的一致性担心了。

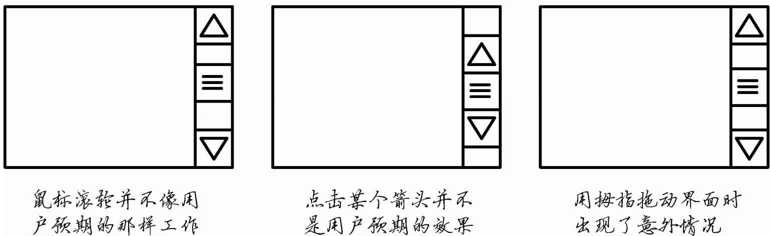
正如拉尔夫·瓦尔多·爱默生所言：

> 愚蠢的一致性为渺小的心灵所敬畏，为卑微的政客、哲学家和神学家所膜拜。伟大的灵魂则与一致性毫无干涉。

因此，让我们扔掉那些愚蠢的一致性，而关注那些真正需要保持一致的地方吧！

16.2　行为一致性

太擅长于识别用户界面元素也会对用户造成些许麻烦：他们会随意把自己大脑中已有的心理模型强加于产品之上。用户界面的某些元素越像用户已知的东西，用户就越期望这些元素能像他们预料的那样运作。



有时你可能会自定义用户界面，这些问题通常发生于这些自定义的用户界面中。因为人们会把自己大脑中已有的心理模型应用到你定义的界面元素中，所以要保证这些元素的行为完全与常用界面元素的行为一致。如果你的设计与用户的预期不一致，那么用户会感到迷惑、沮丧甚至会变得恼怒。①

例如，在 Mac 上运行的 Adobe Photoshop 中的窗口本应该与 Mac 本身的窗口非常相似，但其实却没有。



① 如果你决定重新设计已有的用户界面元素，不要忘了考虑这些元素的可访问性。虽然访问这些元素对你来说可能不算什么，但如果忽略了这些，那么在某些方面有障碍的人可能就无法使用你定义的界面元素。

Adobe 采用了他们自定义的窗口。不幸的是，这样做使得 Adobe 犯了一些行为错误。例如，当用户点击常规 Mac 窗口上的小关闭按钮时，该按钮会变得稍暗一些，以此暗示用户，他点击了该按钮，但 Adobe 设计的按钮并没有这样的效果。Adobe 没有关注到这个细节，虽然只是小小的差异，但它会让用户每次看到关闭按钮时感到迷惑，停顿一小会儿，思考究竟出了什么问题。

如果你不想让自己定义的界面行为类似于常规用户界面元素，那么就要让它从外观上与常规元素有所差异，并且是较大的差异。这样用户就不会把你定义的界面元素当作是常规元素，不会把常规元素的心理模型应用于你自定义的元素，而是建立新的、与传统不一致的心理模型。但要做到这一点并不容易。例如，滚动条、窗口、按钮和菜单的设计通常难免落入俗套。你无法设计出与常用滚动条有天壤之别的方案，能让用户完全不会把自己对滚动条的预期应用到你的设计之上。

最好还是坚持采用常见的用户界面元素和它们标准的行为方式。

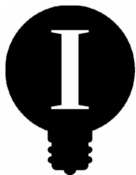
提示

- 人们擅长于识别用户界面元素，所以界面元素没有必要都一模一样。
- 同类界面元素一定要有相似的外观，相同的行为方式，因为人们会尝试把已有的心理模型应用于相似的界面元素。
- 如果你必须创建出与常用界面元素行为不一样的设计，一定要确保视觉上存在差异，这样人们才不会形成错误的预期。你一定不想用一个与常见、相似的用户界面元素不一致的方案来让用户在使用时沮丧不已吧。

延伸阅读

在 37signals 的《把握现实》一书中，作者用精炼的语言描述道：“如果界面的不一致能让你的设计更有价值，那么打破一致性是 OK 的。”

乔尔·斯波斯基在《程序员用户界面设计》中谈到了一致性问题。



第 17 章

可发现性

17

设计师所说的“可发现性”（Discoverability）是指用户是否能发现并使用产品的某些功能。可发现性非常重要，对于用户来说，产品中那些无从发现或无法使用的功能形同虚设，毫无意义。

17.1　哪些功能要易于发现

可发现性通常会涉及权衡问题。使某个功能更易于让用户发现，也就意味着贬低其他功能。因此，你首先要确定的是应该使哪些功能更易于让用户发现，哪些功能不易被用户发现。有时，你甚至可以隐藏某些功能，但前提是你知道那些需要该功能的人能够发现如何使用该功能。

例如，很多浏览器具有一些只有网站开发者才使用的功能。这些功能默认是关闭的，但设计者认为目标用户能够找到这些功能，并知道如何使用。如图 17-1 所示，在 Safari 浏览器中，打开设置窗口，切换到高级标签，选中“在菜单栏中显示‘开发’菜单”就可以使用网站开发功能了。

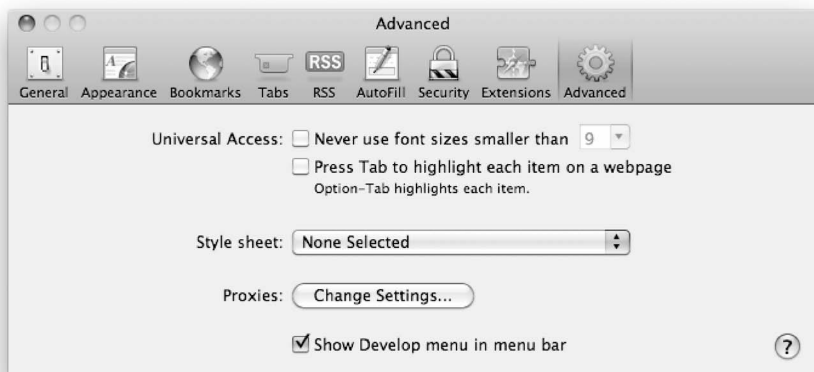
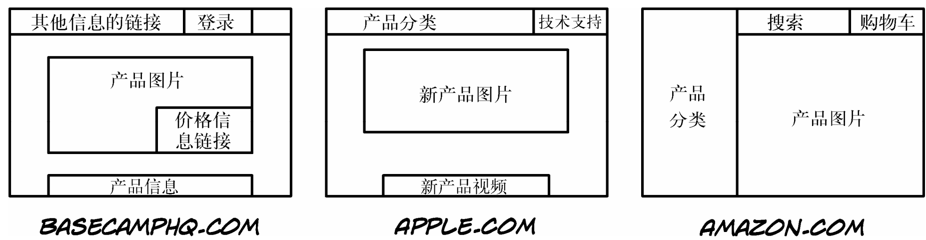


图 17-1　在 Safari 中启用开发者工具

一旦你确定了应该使哪些功能易于被用户发现，应该把哪些功能隐藏在幕后，然后就要确定重要功能的重要程度。不能把所有功能设计得同样明显。

让我们再看一些例子，以下是三个非常受欢迎网站的基本结构：



37signals 的 Basecamp 网站的主页把重点放在吸引新客户上。页面的中间位置显示的是开发计划、价格和功能，其次是诸如“技术支持”和“登录”之类的链接。

苹果公司的主页着重展示最新的产品。导航栏占了一部分空间，让用户能访问到其他产品页面，其他界面元素只占用了很小一部分空间。

亚马逊则重点考虑了网站导航，为产品种类和搜索功能保留了大部分页面空间。因为亚马逊卖的商品种类繁多，所以让用户能轻松找到“导航”和“搜索”功能是网站主页的首要目标。

你首先应该找到产品最重要的功能，然后考虑如何让用户首先发现这些功能。还要考虑某个产品功能的目标用户，如果目标用户有过同类产品的使用经历，那么你可以把这个功能设计得稍微不那么容易被发现，相信用户能够找到它。

在 37signals 和苹果公司的主页中，设计师可能认为面向已有客户的功能可以不那么明显，因为这些人已经知道自己要找的功能在哪里了。潜在的新客户很少浏览公司的网站，因此面向新客户的功能更为重要。

如果某些功能不是十分重要，并且用户可以通过其他方法找到这些功能，那么你就可以让这些功能不那么容易被发现。比如在 iPhone 中，两只手指张开或并拢可以放大或缩小一张图片，这项功能就不太容易被发现，但是不知道这项功能完全不影响你对 iPhone 的使用。另外，如果产品的功能十分出色，知道的人会告诉那些不知道的，那么创建发现该功能的途径就属于应用程序之外的事情了。“用两只手指缩放图片”的功能十分简单，用户见过之后很容易就记住了，并且该功能与其他应用的工作方式相一致。因此，人们在学会使用之后很难会忘记它。

让我重申一下要点，如果存在以下情况，你可以让某个常用的功能不那么容易被发现。

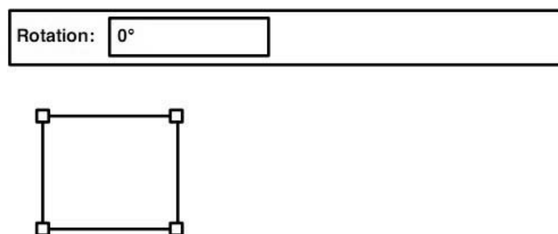
- 虽然人们不知道该功能的存在，但这毫无影响。
- 虽然该功能没有直接显现在用户面前，但用户自己能找出它。
- 该功能十分诱人、简单且经常被使用，人们知道了之后就会记住它。

17.2　何时让用户发现

并不是产品的所有功能都是息息相关的，所以在确定了该让哪些功能易于发现之后，你要决

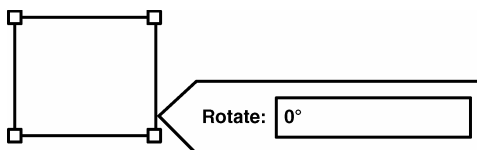
定应该让用户何时发现。

做法之一是使用环境化或模态化用户界面。让我们看一下如何将它们应用于矢量图形编辑器。在使用软件时，用户可能需要旋转某个图形对象，因此你要在工具栏或检视窗口中增加一个“旋转”（Rotation）属性。



这样做需要一个工具栏或检视窗口。你要确保在用户没有选择任何东西的情况下，“旋转”属性是无效的。也就是说，如果无视用户是否选择了对象就增加用户界面元素，就会把界面搞得凌乱不堪。

另一种方法是用户一旦选择了物体，就立刻显示出“旋转”属性。实现这种效果的一种做法是将一系列暂时显示的属性包含在一个小型弹出界面中，只有用户选择了支持这些属性的对象时才将其显示出来。



（当然，究竟这样做是否有效，我们还得让用户对这些方案进行测试。）

17.3　如何让用户发现

至此，我们已经考虑到了产品的重要功能，也考虑了这些功能在何时显得十分重要。现在，我们看一下如何让用户发现这些功能。

空间属性

你可以利用诸如大小、位置、形状和颜色之类的属性，让程序中的某些元素更容易（或更不容易）被发现。尺寸越大，就越容易被发现。把某些东西放在屏幕或窗口的左上角或左下角更容易被发现。[①]某些颜色（大部分明显的红色）能让事物显得更加重要。如果你认为某些东西稍微

① 雅各布・尼尔森称之为“F 样式”（F-Pattern），因为在视觉跟踪研究中，测量人们浏览网站时的关注点所成的图像类似于一个巨大的、附加在屏幕上的 F。关于此研究的例子，参见 http://www.useit.com/eyetracking。

不那么重要，但仍然要让用户能够发现，你可以用相反的手法让它变得不那么明显。[①]

颜色

目前的设计有在用户界面中少用颜色的发展趋势。例如，苹果的 iTunes 已经用单色图标取代了以前的多颜色图标。

从可用性的观点来看，使用颜色是有其益处的。颜色能让事物更容易发现。我们的大脑真的很擅长执行诸如“在屏幕上找出绿色图标”之类的任务。科林 · 威尔在《信息可视化》中提道：颜色会被“提前关注到”，其含义是，我们在自己意识到颜色之前就已经发现它了。换句话说，在观察用户界面时，我们能很快、很容易地发现那些有特定颜色的界面元素。

你可以使用颜色（比如通过使用颜色让图标产生差异），但不能仅依赖于颜色来提升设计效果。另外，不是所有人都能精确地发现颜色。

用户期望

一旦人们已经使用了你的产品一段时间，他们自己就会知道哪些部分比较重要。因此，界面设计要保持一致，每一屏采用相同的布局方式。不要在这一屏内把重要的东西放左边，到了下一屏又放在右边。

与此类似的是，如果用户已经建立了产品工作原理的心理模型，他们可能会有非常强烈的观点，认为某些东西应该在某些特定的位置。通常，你可以利用这种预期来让某些内容更加容易（或不容易）被发现。关于这方面的更多内容，参见第 9 章。

搜索

如果用户不能很快找到某些东西，他们可能会求助于产品的搜索功能，只要他们能找到这个功能。为了帮助用户使用搜索功能，你需要做到以下三件事。

- 提供搜索功能。
- 保证用户能很容易地找到搜索功能。
- 确保搜索功能可以返回有用的结果。

通常，最后一点是最难做到的。幸好你可以从第 7 章中的“卡片分类”中找到一些相关信息，这些信息能帮助你找出该用什么样的术语描述搜索结果。用户当时使用了什么样的词汇？如果用户在搜索引擎中输入了这些词，是否能得到预期的结果？

另外一种改进搜索结果的方法是，关注用户在可以使用搜索功能时所选用的搜索方式。特别

① 在用来创建内容的应用程序中可采用中性色。我们不想让一个彩色的用户界面影响用户（在图片编辑工具中）观察他们所编辑的图片。

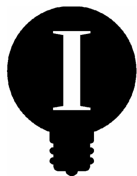
要注意那些返回空结果或返回很少结果的搜索词汇。还要注意这种情况：用户使用了搜索功能，网站也返回了搜索结果，但用户却没有点击其中任何一个。

动画

动画是吸引用户注意力最强大的工具。一定要巧妙、谨慎地使用它，对那些长时间显示的内容坚决不要使用动画。关于使用动画的更多内容，请阅读第 15 章。

提示

- 首先决定哪些内容应该让用户易于发现，哪些内容需要被隐藏在某些地方。然后为前者分配重要程度，并根据重要程度进行设计。
- 谨记，并不是产品的所有功能在同一时间内都可用。不同的内容需要在不同的时间被用户发现。
- 使用视觉布局、出色的搜索功能和动画（在少数情况下）来确保内容易于发现。



第 18 章

不要打扰用户

心理学教授米哈里·契克森米哈赖在他的著作《幸福的真意》（*Flow: The Psychology of Optimal Experience*）中说道，当人们完全投入到自己所做的事情中时，会进入一种全神贯注的状态，他称之为“流”（flow）。当人们谈到自己处于某种状态时，他们通常指自己毫无保留、一心一意地沉浸在某一活动中。

通常，我们需要一定的时间才能进入最佳精神状态，但却非常容易被拉回到常态。或许我已经完全沉浸于撰写本章内容的工作之中，但突然间电脑发出“叮”的一声响。我想道：“新邮件！”然后就会花费五分钟时间来读新信息。这样，我就跳出了那种最佳写作状态。

打扰用户的代价是非常高的。用户不仅要花时间来处理自己所受到的影响，还要花时间回到自己受打扰之前的状态——如果可能回去的话。微软最近的研究表明，微软员工在受到新邮件或即时信息的打扰后，平均需要花费 15 分钟才能回到受打扰之前的状态，继续执行任务。人们很容易走神，且常常要花费较长的时间才能从走神的状态下恢复。[1]

因此，打扰用户会浪费他们的时间。用户也会很生气，甚至有时会因此变得粗鲁。毕竟，干扰是外界影响因素，它会打扰到用户，使他们停止当前正在进行的工作。

在任何可能的情况下都不要打扰用户，以下是做到这一点的方法。

18.1　帮助用户作决定

如果你可以帮助用户做出某个决定，那就去做吧。不要仅仅为了让用户确认某个操作而打扰用户。

例如，每次把读卡器插入到 Windows 系统的电脑时，我都会看到以下信息（见下页图）。

最后一个选项是利用读卡器里的 4G 存储卡来做备份，我有多大的可能会执行这一操作？有多少人会点击这个窗口中的第一选项“用 Windows 导入图片和视频”？为什么系统不默认打开

① 史蒂夫·洛尔（Steve Lohr）在《纽约时报》（The New York Times）的一篇文章中对类似的研究进行了总结，参见 http://www.nytimes.com/2007/03/25/business/25multi.html。

存储卡窗口，而是用一堆大部分人甚至都不去读的选项来打扰用户呢？



我使用 iPhone 的时候经常出现以下情况：在某个应用中，我选择了某个特定的选项，例如在视频播放器中点击了“播放”，或在 Twitter 的客户端中点击了某个链接，但该应用并没有直接执行我的输入，而是扔出了一个包括其他选项的界面。图 18-1 即是两个类似的例子。

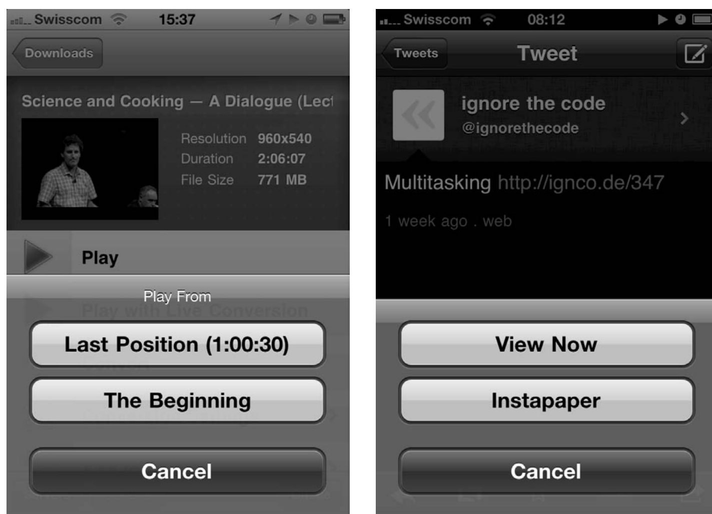


图 18-1　不要让我去选择

Air Video[1]在用户点击了“播放”之后询问用户想从哪里开始播放文件。Twitter 用户端 Osfoora[2]在用户点击了链接之后询问用户是想浏览网页还是想把该链接添加到 Instapaper 中。

① Air Video 是面向采用 iOS 系统设备杰出的流视频播放软件，更多内容参见 http://www.inmethod.com/air-video。
② Osfoora 是一款非常简洁、漂亮的 Twitter 客户端应用，参见 http://www.osfoora.com。

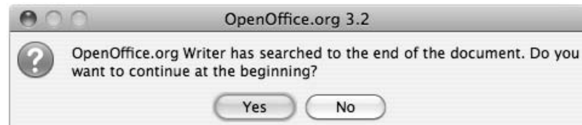
以上这两种对用户的打扰都是完全没有必要的。应用程序应该帮助用户做出选择，执行用户最有可能选择的选项，并向用户提供可以改变选择的方法。

例如，在用户选择了播放某个影片之后，从上一次播放的位置开始播放就是了，这是用户最有可能选择的方案。当然如果不是的话，用户会跳回到影片的开始处进行播放，这很容易做到。同样，如果他点击了某个链接，那么就直接打开这个链接，这是用户最有可能选择的选项，同时提供把已打开页面添加到 Instapaper 的方法。

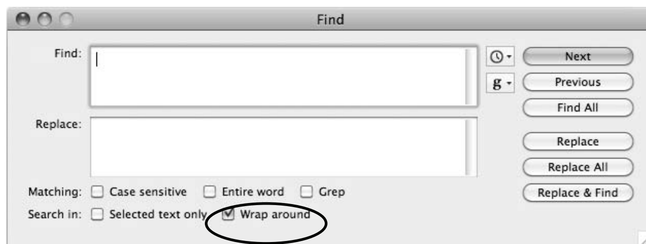
18.2　提前作决定

某些情况下，你不得不让用户自己来作决定。此时，最好让用户提前一次做完所有的决定，然后不再被打扰。例如，你最好让用户在开始安装软件时就输入软件的许可证信息，而不是在安装过程中间停下，让用户来输入软件许可数据。

以下是另一种不必要的干扰，来自于 OpenOffice.org：



不要在用户执行搜索时打扰用户，只需在搜索对话框中加入选项，以下是 BBEdit 的做法[①]。



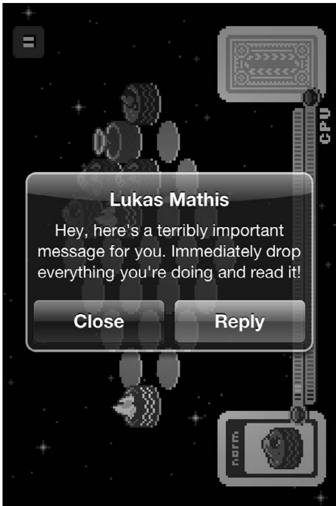
或者，最好是根本不询问用户。默认选择搜索环绕文本，在搜索执行到这类内容时显示一个简短的动画，这样用户可以在不想选择环绕选项时停止搜索。

18.3　只在做出紧急决定时才打扰用户

永远不要仅仅为了告知用户发生了某些事情而打扰用户，如下页图所示。[②]

① 参见 http://barebones.com。

② Mozilla 公司 Firefox 项目前创意设计总监阿扎・拉斯金说道，在不涉及作决定的情况下打扰用户是“毫无效率的行为”，因为用户只能做一件事情。不管如何应对类似的干扰，用户永远都不会给电脑带来任何新信息。更多内容参见 http://barebones.com。



在这种情况下，用户实际看到的东西如下。

我正在强迫你停下当前的工作！现在看看这个吧！
你收到一条新文字信息。
我每天都会收到几十条新文字信息。别再打扰我了！

如果信息不是特别重要，那么就不要显示它。如果它确实很重要，但不需要即时的处理，那么请用不打扰用户的方式显示该信息。例如，你可以用一种非模式化[①]、无干扰的方式来通知用户。以下是 HP 的 webOS 向用户显示新信息的方式。



① 如果是非模式化的，那么信息不会打扰到用户正在做的事情。关于模式的更多内容，参见第 20 章。

当收到新信息时，带有小信息图标的黑色色带会从屏幕底部悄悄地浮现出来。触摸黑色带会显示更多关于信息的内容。触摸信息本身，会在应用中打开它。

打扰用户是非常不礼貌的行为

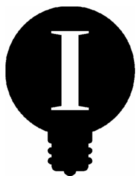
除非你实在是没有其他选择，不得不要求用户输入操作，除非你立刻需要某些信息，否则，不要打扰用户，那是非常不礼貌的行为。

提示

- 不要打扰用户。打扰用户会让他们不高兴，降低其工作效率，这本身也是不礼貌的行为。
- 帮助用户作决定，而不是向他们提问题。
- 如果不能帮助用户作决定，一次问完所有需要的问题，而不是每次遇到新问题时都打扰用户。
- 如果你必须告诉用户某些事情，保证尽量不打扰用户，这样他们才能按照自己的时间安排来处理这些事情。

拓展阅读

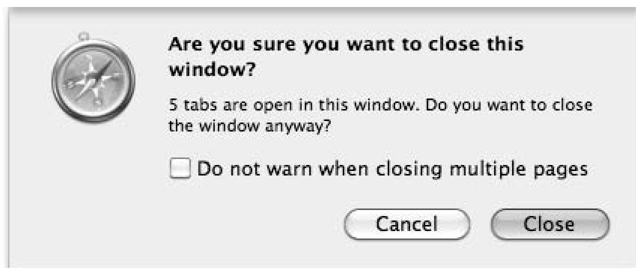
米哈里·契克森米哈赖的著作《幸福的真意》解释了人们在关注于某项任务时不受打扰的重要性。



第 19 章

用“撤销”取代对用户的干扰

你是否曾遇到过这种情况？在使用电脑时偶然做出了不正确的操作，电脑对此向你发出了警告，但你却关掉了警告信息，因为你在想：“我知道自己在做什么。让开，电脑！”



我就经常遇到这种情况。我们对类似的警告信息对话框都是司空见惯，不以为然，常常都不去关注这些信息。“你确定要……？”是，是的，如果我不确定，就不会点击相关的按钮，不是吗？“你真的想要删除掉这些……吗？”是，是的，我真的想这样做。“你将无法恢复……”是！是！是！不，等等，我刚才做了什么？

后来，我们会条件反射似地关掉警告信息对话框。心理学家称这种现象为（对反复出现情况的）习以为常（habituation）。大多数时候，对话框都是让人讨厌的干扰因素，是背景噪声，因此我们习惯了忽略它。当我们真正做出了某些错误的操作，在意识到自己的失误之前，我们已经点击了“OK”。

警告信息是把责任推卸给用户的最佳借口。毕竟，他们应该阅读警告信息，不是吗？然而，他们确实无可避免地会出错。

幸好，大部分时候，我们可以提供比警告对话框更好的解决方案。

你想实现的效果

重要的警告信息
亲爱的用户，你正在执行危险的操作。请稍微考虑一下，你真的要这样做吗？
谢天谢地，感谢你阻止了我这样做！　　是的，我确定要这样做。

用户实际看到的效果

令人讨厌的干扰
虽然你正在处理某些重要的事情，我仍然要打断你，因为……
我管你发生什么！别打扰我，正忙着呢！

19.1　允许用户撤销自己的行为

“撤销”操作不用强迫用户处理不断出现的警告，而是提供了一种简单易懂的解决方法。除非用户偶然做出了某些自己不想做出的操作，否则，撤销是不可见的。一旦发生意外，用户可以很容易地返回到之前的状态。

关键在于，人们要能够信任“撤销”的作用。这意味着，你必须记录尽可能多的操作行为，提供多层次、“有一定深度”的撤销操作，能够允许用户撤销更多操作，而不仅仅是最近执行的。如果“撤销”操作不能提供可靠的结果，或返回以前状态的程度不够，你的应用程序将失去用户的信任。用户会犹豫是否要去探索产品的新功能，在使用已知功能时也会畏首畏尾，最终会在使用程序时变得不高兴。

信任

让用户信任你的程序是最重要的任务之一。可靠的“撤销”功能仅仅是一个方面，你还要让程序的行为可预测，性能稳定。

我曾处理过一个带有严重漏洞的产品，它的某项功能会损坏用户的数据。虽然我们很快修复了问题，但可用性测试显示，在第一次发生那个问题后时隔多年，那些受该问题影响的人仍然避免使用产品的那项功能。

用“撤销”操作减少警告对话框的数量能带来额外的好处：用户不耐烦地点击关闭警告对话框的可能性会减小，因为他们不会像以前那样对这些对话框习以为常。

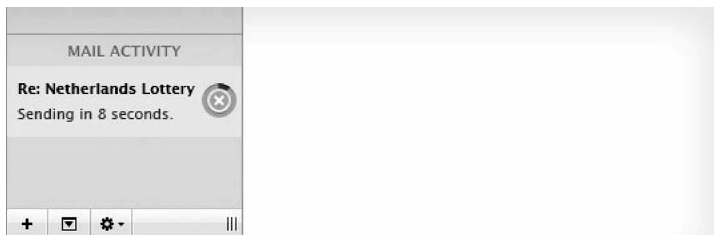
19.2　临时撤销

在某些情况下，由于技术原因可能会无法实现撤销功能。你无法让用户撤销正在发送的邮件，因为一旦发送，就超出了你的控制范围。同样，你可以删除某条曾经发送出的 tweet 信息、某篇曾经发布的博文，但这和阻止 “发送”和“发布”操作不可同日而语。

解决这一问题的方法之一是延迟执行操作，允许用户临时撤销操作。不立刻发送邮件，而是在几秒钟内显示一个“撤销”按钮。如果用户撤销操作，那么就不再发送邮件。否则就移除“撤销”按钮，发送出邮件。Gmail 在它的网页界面中实现了这一功能：



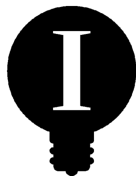
图形和交互设计师克莱顿·米勒称这种功能为“延迟的被动确认”，因为用户没有阻止正在进行的操作，就等于被动确认了该操作。以下是他制作的某桌面程序中该功能的实体原型。[1]



提示

- 在用户做出某些具有潜在危险的事情之前提出警告并不会奏效，因为用户会忽略这些警告信息。
- 允许用户撤销他们的操作，这样才能避免发生意外。
- 如果技术不支持“撤销”操作，至少要延迟具有潜在危险的操作，这样即便用户发出了命令，也仍然能够阻止事故的发生。

① 关于他更多的想法，参见 http://iuface.net/7u8。



第 20 章

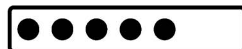
模　　式

1985 年，在为苹果公司撰写名为《苹果机技术内幕》（*Inside Macintosh*）的 Mac 开发者文档时，卡罗琳・罗斯认为“模式”（mode）特别重要，应该在第一卷卷首谈到。她写道：“模式是应用程序中用户必须正规地进入或退出的一部分，在起作用时，它限定了可执行的操作行为。”

大致说来，模式是应用程序或独立窗口所处的状态，它改变着应用程序或窗口对用户输入的反应。模式可能是可用性问题的根源所在。罗斯写道：“人们在现实生活中通常不以模式的方式工作，因而面对计算机软件中的模式问题时，这更强化了他们对于电脑非自然、不友好的印象。”另一个关于模式的问题是：如果人们不知道某个应用程序或窗口所处的模式，他们的输入会产生意料之外的结果。

Caps Lock 键或许是最常见的关于模式的例子。该切换键是否处于激活状态能够改变用户界面对用户输入信息的反应。





小测试：在以上两个密码输入框内，其中一个的内容是在大写切换键打开的状态下输入的，是哪一个呢？

用户只能在其中的一个模式下登录。在另外一种模式下，即便使用相同的输入内容，用户也无法登录。

20.1　隐性模式

杰夫・拉斯金在《人本界面》（*The Humane Interface*）一书中说道：“模式是界面中引发错误、迷惑、不必要的约束和复杂性的重要根源所在。”

如果人们不知道自己处于何种模式，也就是说，如果模式是隐性的，那么模式就会令人迷惑。例如，Caps Lock 键开启时一个小灯会亮，但这很容易被用户忽略。

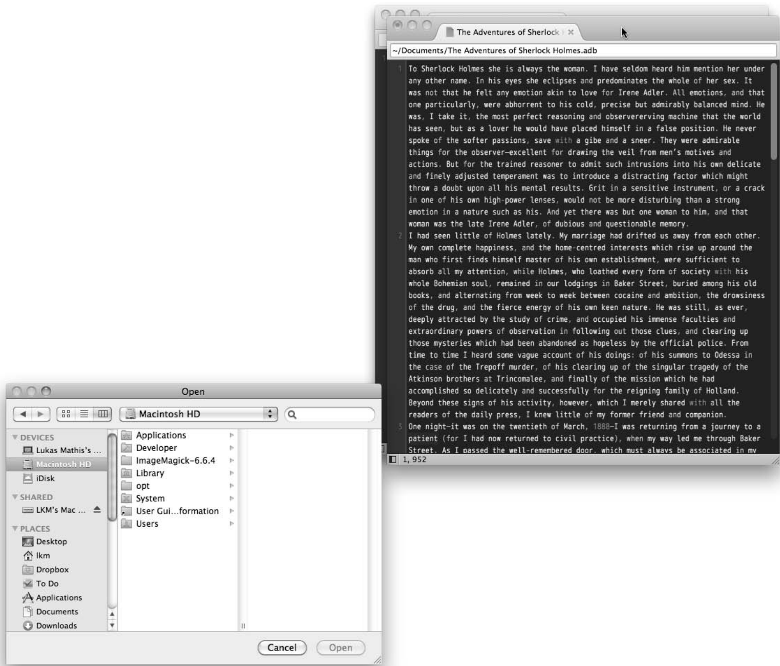
某图像编辑软件中正在启用的工具是另一个隐性模式的例子，见下页图。



小测验：在以上两种情况的某一情况下，点击并拖动鼠标会创建一个矩形；在另一种情况下，会创建一个圆。但究竟在哪种情况下会创建矩形，哪种情况下会创建圆呢？

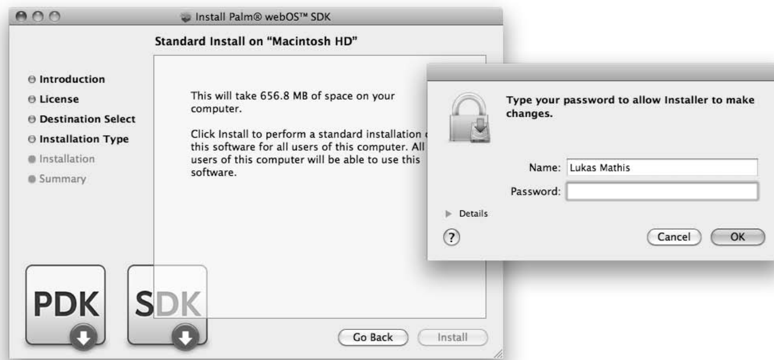
由于不知道用户界面正处于哪种模式之下，人们的操作可能会得到意料之外的结果。用户无法正确登录，或者，在想绘制一个矩形时，图像编辑软件却画出了一个圆。

即便用户忘记已经启用了某种模式，也一定要清晰地表明当前的模式。例如，在 Mac OS X 中选择“打开文件”菜单，系统会打开一个模式对话框。用户在选择“打开文件”之后能迅速了解到界面所处的模式，但如果他喝了杯咖啡，五分钟后返回到自己的电脑旁边，可能就会暂时忘记电脑仍然处于“打开文件”模式之下。



此时，点击文本窗口只能让电脑发出“哔哔哔”的声音，用户暂时也不知道为什么会这样。当然，在这种特定情况下，模式并不是必要因素。为什么“打开文件”对话框会卡住整个应用程

序呢？没有道理要这样啊？但是，在其他情况下，仅仅去掉模式是行不通的。例如，程序安装器会要求用户输入密码。



如果去掉“要求用户输入密码”这一模式，用户能够在密码窗口依然打开的情况下激活安装程序，但却不能继续安装程序。与无法激活安装程序窗口相比，这样做会让用户更加迷惑不解。你不能总是通过去掉模式来解决问题，必须让用户清晰地了解当前正在进行的事情。

让模式显性化的一种方法是提供视觉反馈，暗示某一模式处于激活状态。用户界面设计师比尔·巴克斯顿（Bill Buxton）在题为 The Prevention of Mode Errors Through Sensory Feedback[①]的论文中描述实验结果时写道：“无论是对于新手还是专家用户，有视觉反馈比没有能极大地减少用户所犯的模式性错误。”在测试视觉和动觉反馈时（后者基于对用户肢体动作的感知——例如在手机处于静音模式时，你能感觉到振动），他进一步评论说：

> 测试发现，无论测试对象是否有过产品使用经历，视觉和动觉反馈都能带来益处。（……）因此，即便大部分专家用户声明，他们已经习惯把产品模式记在大脑中，这两种反馈也能极大地减少他们所犯的模式性错误。

如果静态视觉反馈无效，那么动画或许能有所作为。正如在第 15 章中所讨论的那样，动画比静态视觉元素更难以被忽略。

但是，视觉反馈并不是最好的解决办法。巴克斯顿的研究表明，动觉反馈更有效。或许你会想，在不为用户的电脑添加特殊硬件设备的情况下，该如何才能实现动觉反馈？你可以把键盘上的按键作为模式激活器。例如，在 Mac OS X 下拖动一个文件，Finder 将移动文件；但如果同时按下 Alt 键，Mac OS X 会切换到“拷贝”模式。用户必须接触并按下 Alt 键才能保持住这一模式，这就是一种动觉反馈。持续的触觉反馈能告知用户，他仍然处于拷贝模式。这种短暂的、不间断地按下按钮才能激活的模式称为“准模式”（quasimode）。我将在本章末对此进行讨论。

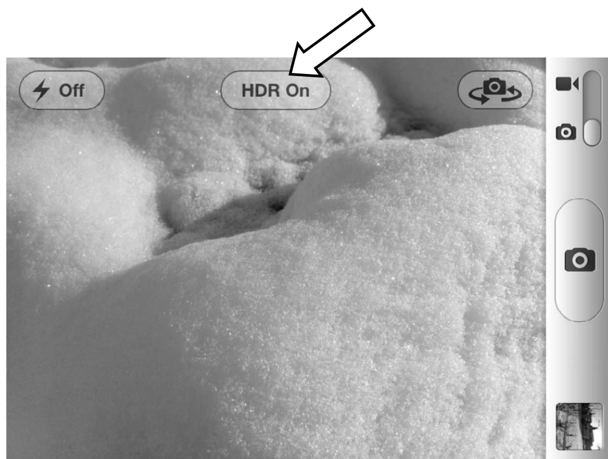
① 参见 http://www.billbuxton.com/ModeErrors.html。

阿扎・拉斯金在他的文章 Visual Feedback and How Modes Kill[①]中提到了另外一种方法：

> 视觉暗示是一种能把你的注意力集中到系统状态的方法，但正如巴克斯顿的实验所证实的那样，它有可能会失效。使用基于声音的反馈可能会更有效，因为虽然你可能会忽略某个视觉提示，但你无可避免地会听到音频提示（每个听过小马里奥哭声的"超级马里奥 2"玩家都知道这一点）。

你最好尝试解决这一问题的各种办法，然后进行可用性测试，看看哪一种方法最有效。

但即便设计师知道必须让模式显而易见，并找到了可行的办法来实现这一目的，用户仍然可能会对设计结果迷惑不解，因为模式的表现方式不明确。以下是一个来自 iPhone 的例子，它的内置摄像头应用程序有一种 HDR 模式。[②]



但"HDR On"究竟是什么意思？是表示我可以通过触摸按钮打开 HDR，还是表示它目前处于开启状态，触摸按钮会关掉它吗？杰夫・拉斯金在《人本界面》中推荐使用复选框或单选按钮来表示模式的开启。

☒ HDR

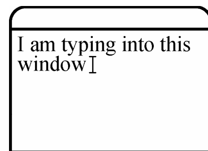
这样的图标就非常清楚。

① 参见 http://www.azarask.in/blog/post/is_visual_feedback_enough_why_modes_kill/。

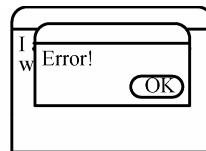
② 高动态光照渲染（High Dynamic Range）为图像结合了不同的曝光方式，相比正常摄像头，它能产生带有更好动态渲染效果的图像。

20.2　意外模式

对话框是一种特殊的模式。与隐性模式不同的是，对话框通常比较明显，至少在第一次弹出的时候比较明显。对话框常常出乎用户意料之外，因此它会引发与隐性模式一样的问题：用户界面没有实现用户想要的结果，因为它不在用户期望的模式之下。



用户期待键入的文字出现在文档中



但意料之外的对话框“吃掉了”用户键入的文字

对话框不仅能把用户界面转到用户意料之外的模式，而且让用户的输入也可能被误认为是针对模式化对话框的，而不是针对先前处于激活状态的窗口的。（这种情况被称为“偷换焦点”，因为它把电脑的焦点从用户希望的位置转移到了其他地方。）这意味着，用户可能会偶然地放弃某个模式化窗口，或接受某些原本要拒绝的操作。Enter 键可能不会把光标移到下一行，而可能是卸载某个网络硬盘或是删除某个文件，因为这是模式化窗口中默认按钮的操作。

20.3　难以退出的模式

下图是我的闹钟，为了在闹铃关闭的模式下更改时间，我必须按下最上面的按钮。这样会把闹钟切换到“更改闹铃时间”模式。在这种模式下，最上面的按钮可以更改闹钟设定的时间。因此，如果我不能通过再次按下这个按钮来退出这种模式，那么如何才能退出该模式呢？



用户通常不能明显看到该如何退出被激活的模式，这就会引发可用性问题。PC 键盘上的 Insert 键是另一个具有相同问题的例子。一旦用户碰触到了 Insert 键（可能是偶然性的），就激活了改写模式，而如何返回到“正常”模式却完全不明显。

如果要使用模式，一定要让用户明确知道该如何退出模式。

对于这个闹钟谜题，事实上你不能主动离开“更改闹铃时间”模式。你可以按下所有想按下的按钮，但实际上，这只能让你一直陷于该模式的泥潭中，因为只要用户停止疯狂地、任意地按下按钮试图离开这一模式，闹钟就会在几秒钟内自动退出“更改闹铃时间”模式。

20.4　模式并非一无是处

模式会引发许多问题，难怪它会臭名昭著。但实际上，它本不应落得如此下场。把用户界面设计得完全非模式化会增加其复杂性。在需要的时候，模式可以用来表明产品的功能，而非模式化的界面在任何时候都必须提供大量可能的用户操作。

只有那些隐性的、意外的、难以退出的模式才是不好的设计。如果用户是有意转换界面的模式，并且模式在处于激活状态时是明显易见，同时可以让用户时刻知道该如何退出某一模式，那么在界面设计中使用模式并没有任何过错。

记住这些基本原则，模式才能改进用户界面，而不是让界面更具迷惑性。

20.5　准模式

杰夫·拉斯金称之为“准模式”的模式是指那些只要用户明确地保持激活就能存在的短暂模式。罗斯在《苹果机技术内幕》中称这种模式为“弹性加载模式”。与 Caps Lock 键对应的准模式是 Shift 键。Shift 键并没有把电脑切换到一种永久模式，而是引入了一种瞬时模式，只要用户按下该键就能激活。由于用户必须明确地保持准模式的激活状态，该模式也就不可能是隐性的、意外的，或是难以退出的模式。

罗斯在她的书中提到的另一种准模式是按住鼠标键，只有用户持续按住鼠标键才能激活这一模式。电脑在鼠标键按下时行为变得不同，但用户永远不会为自己当前是否正按着鼠标键感到迷惑。

如果需要使用模式，考虑一下准模式是否能解决某个用户界面问题。

提示

- ❑ 如果设计不当，模式会使用户迷惑不解。
- ❑ 模式非常有用，它能避免让产品充满用户界面元素。

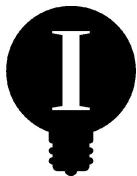
- 为了让模式良好地发挥作用，它应该是显性的、可预期的、能够轻易退出的。
- 在可能的时候使用准模式，而不是完整模式。

延伸阅读

很多用户界面设计相关的书中都会提到模式。例如，《交互设计精髓》就在多处提到过模式。杰夫 · 拉斯金的《人本界面》中有一章介绍模式。阿扎 · 拉斯金撰写了一篇非常棒的关于模式的文章。[①]

虽然《苹果机技术内幕》一书只浅显地介绍了模式，但大体来说也值得一读。

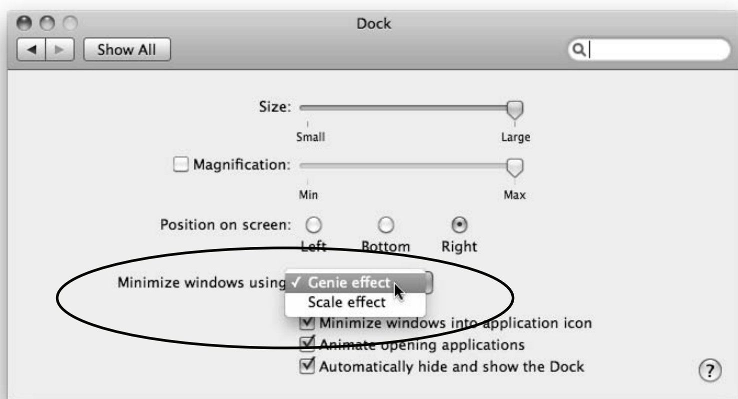
① 参见 http://www.azarask.in/blog/post/is_visual_feedback_enough_why_modes_kill/。



第 21 章

用你的观点代替偏好设定

如果你不确定该选哪个设计方案，最好让用户来做决定。对于 Mac 窗口缩小到 Dock 这一过程，某些人想让窗口优雅地飘进去，而其他人则更喜欢让窗口只是快速地线性缩小。为什么不同时实现这两种效果呢？



设定窗口中所有不必要的复选框都是你留给用户的设计选择问题，而用户比你更不擅长于做出选择。你是最有可能做出正确选择的人，不要让对产品不甚了解的人代替你做决定，他们做出的选择可能比你自己的选择还要差。

设计师兼开发人员麦克·朗德（Mike Rundle）对此说道：[①]

> 如果你通常在 A 和 B 两个选项中选择 A，为什么不直接把 A 设定成主要的、不可更改的选项呢？如果你认为 A 是最佳选项，为什么还让别人选择 B 呢？（……）
>
> 史蒂夫·乔布斯实现了他自己想要的产品，因为他知道自己创建的是非常优秀的产品。你也应该如此！

① 关于他的文章，参见 http://flyosity.com/iphone/kill-the-settings-build-opinionated-software.php。

你经过了专业训练，适合来做这些选择，也拥有足够的信息做出正确的选择。而你的用户则不具备这些。

我讨论的是什么？

我并不是说你在设定窗口中看到的所有东西都不好。我讨论的是偏好设定，即某事物有两种运作方式，但某些人更喜欢其中一种。让我来定义一下本章中将用到的术语：

设置（settings）	用户对你的产品功能或行为所能做出的全局性更改
配置（configuration）	产品正常工作所需的设定，比如屏幕分辨率或网络设定。如果你正在创建一个Twitter客户端，你应该允许用户对软件进行配置，显示他自己的Twitter信息
偏好设定（preferences）	改变产品行为的设定，通常不是严格必需的，但某些人或许会比较喜欢这些选项，例如Apple菜单或Windows开始菜单上显示的最近使用程序的个数，Mac OS X中滚动条上箭头的位置等
个性化设定（personalization）	纯粹实现某种视觉效果的设定，不会改变产品的实际行为，比如桌面背景

当我说“你应该避免使用偏好设定”时，我并不是指配置（毕竟，它是产品正确工作所必需的设定）。

我也并不是在说个性化设定。虽然产品正常运行并不一定必须个性化设定，但它能极大地提升用户的使用体验。如果你的手机允许把宠物狗的照片设为锁屏界面，你会更加喜欢它。

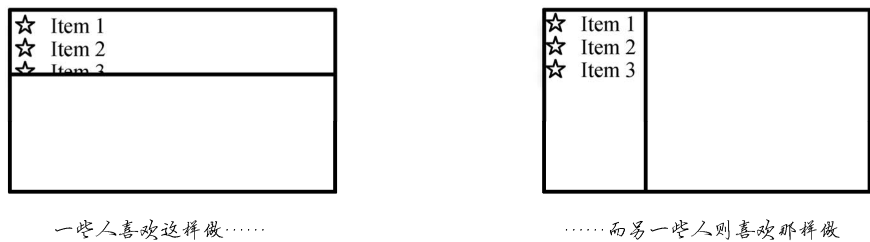
应该把某个选项当作产品的配置、偏好设定还是个性化设定？这个问题通常没有完全明确的答案。但从这三个类别来考虑产品的设定能帮助你做出更好的选择，知道应该保留什么，放弃什么。

21.1　为什么偏好设定不好

偏好设定会以各种方式为你的产品带来完全没有必要的复杂性。首先，人们会进入到产品的设定界面，寻找他们需要更改的东西，任何你提供的不必要的选项都会让他们更难找到自己想要找的东西。他们不得不一一浏览每一个偏好设定，这让所要找的目标更加难以发现，也会让整个寻找过程更加令人沮丧。

其次，偏好设定会引入新的模式，把你的产品变得不一致。每个偏好设定都是一种模式（参见第 20 章）。如果你曾用过一个喜欢摆弄操作系统偏好设定的用户的电脑，你很快就会发现自己常常被某些东西搞得恼怒不已，因为产品的功能总是处于你意料之外的模式。例如，当用户插入一张 CD 时，某些电脑的桌面会弹出一个 CD 图标，而有些电脑的这项功能则处于关闭状态；在某些电脑中点击滚动条会让屏幕显示内容滚动一屏内容，而有些电脑则会滚动到你所点击的位置。你很容易习惯此类状况，但如果你使用的是一台拥有不同偏好设定的电脑，当电脑每次处于你意料之外的模式时，你都会感觉到些许挫败感。避免使用偏好设定，就能避免让用户陷入这种境地。

再次，偏好设定会让那些做出错误选择的人感觉你的产品很难用。某些人喜欢把未阅读的RSS 条目放在当前打开条目的左边，而有些人则喜欢放在上边。



一些人喜欢这样做……　……而另一些人则喜欢那样做

现实中总有一种对大多数人都有效的方案，你要做的就是找出这个解决方案。你可以对产品的不同配置进行测试，来寻找最有效的方案，但你的用户却做不到这一点。

最后，把选择权留给用户也就意味着你不得不为用户提供产品的多种不同使用方式。如果你不让用户来做选择，而是自己选出一个选项，那么就不必在其他选项上浪费时间，也就有时间把所选择的那个选项做到最好。并且，当有问题发生时，偏好设定会加大修复问题的难度。每一项无意义的偏好设定都会增加产品的代码路径。如果有错误发生，用户能够改变产品运行方式的方法越多，重现问题并提供及时帮助的难度就越大。

21.2 如何避免偏好设定

对用户说“不”（参见第 24 章）。有些人真的很喜欢较小的字，他们甚至会给你发邮件，要求改变文字的大小。但这并不意味着为产品增加调整字体大小的偏好设定是最好的选择。你可能会讨好某些用户，但却会让其他所有人都感到不方便，你还必须为此让窗口支持各种大小的字体。

对自己说“不”。偏好设定听上去可能是个不错的选择，通常这是最快捷的方法。为某些东西进行偏好设定，你就不必亲自做选择了，但这对用户来说却是在帮倒忙。还记得本书最开始处（第 3 章）关于用户的讨论吗？对于每个偏好设定，请考虑以下问题：它以怎样的方式为用户提供帮助？它能帮助用户实现哪些以前无法实现的目标？如果你找不出明显的答案，请放弃这项偏好设定。

与同类产品保持一致。如果你无法在两个选项中做出抉择，而用户可能使用过类似的产品，那么请参考这些产品，这样用户就能把自己已有的知识应用于你的产品。

进行可用性测试。如果某个问题有两个解决方案，你不知道该选哪一个，那么创建纸质原型，对这两个方案进行测试（参见第 11 章）。或者同时采用这两个方案，然后执行 A/B 测试（参见第 33 章）。从这些测试中得到的反馈和数据甚至能帮助你找出新的、比你最初的想法更好的方案。

持有主见。如果你面前有两条路可以走，那么选择你更喜欢的那条路吧。你更喜欢用哪个方

案？自己持有主见是不错的。如果你想要讨好所有人，那么将会没有一个人感到高兴。

给出隐性的偏好设定。不要让用户专门设定产品行为，而是要记住用户上一次的操作；不要让用户设定窗口的默认尺寸，而是使用用户上一次缩放后的尺寸作为默认窗口尺寸；不要让用户在文件共享程序中设定自己喜欢的通信协议，而是把他们上次使用的协议作为默认设定，在需要的时候允许他们进行更改；在用户打开某个程序时，不要让他们设定自己想要看到“新建文档”或“打开文档”之类的对话框，而是再次打开他们上一次用这个程序打开的文件。

21.3　如果无法避免偏好设定

如果你真的无法避免偏好设定，一定要确保大部分用户认可每个偏好设定的默认选择。大部分人都会使用默认值，从不进行更改。可用性专家雅各布·尼尔森指出：[①]

> 很多以前的研究，包括我自己的研究都表明，搜索结果中少数前几项的点击率占比最高。其中，第一项的点击率极大地超过了其他几项。（……）在用户界面设计中的很多地方，用户都依赖默认设定。例如，他们很少会使用能实现华丽效果的功能，因此一定要重点优化默认的用户体验，因为这是大部分用户的选择。

赋予每个偏好设定合理的、统一的标签术语。不要使用用户不熟悉的标记或行话。

不好：　☒ 下刷显示更多

较好：　☒ 下刷显示所有窗口

避免使用否定式标签。复选框应该用来启用某些选项，而不是关闭。

不好：　☒ 禁止自动登录

较好：　☐ 自动登录

在使用手册中说明各个偏好设定的作用，并用工具提示向用户提供额外的解释。如果某个偏好设定因故不起作用，要通过工具提示向用户解释原因。[②]

提示

- 产品的设定可分为配置（产品正常工作所需的设定）、个性化设定（纯粹的视觉变化，并不是必需的设定，但它能让用户感觉到产品是属于他的）和偏好设定（功能性的改变，并不影响产品的正常工作）。

① 关于完整的文章，参见 http://www.useit.com/alertbox/defaults.html。

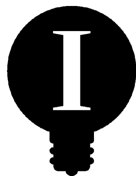
② 某些操作系统不支持为无效的用户界面元素显示工具提示，但我确信你能找到其他相当明显的方法，来向用户解释为什么某个特定的偏好设定无效。

- 避免使用偏好设定，因为它会引入不必要的模式。
- 不要支持偏好设定，而是实现最恰当的默认产品行为。

延伸阅读

设计师兼开发人员麦克·朗德曾撰写过关于本章主题的文章。[1]

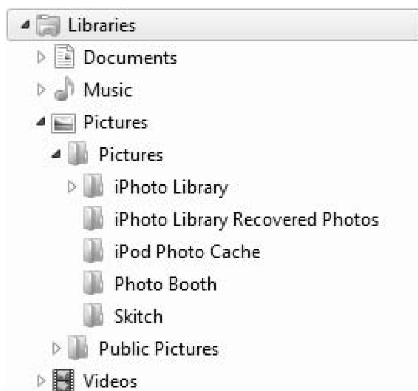
[1] 参见 http://flyosity.com/iphone/kill-the-settings-build-opinionated-software.php。



第 22 章

层级结构、空间、时间以及我们对世界的看法

如果允许，电脑通常支持用户以层级结构的形式组织管理数据。PC 中的文件系统就是最明显的例子。



层级结构适用于很多东西。人类已习惯于层级结构，因为它随处可见。家族就是由层级结构组成的（族谱）。我们关注一下动物就会看到，各个物种在进化层级结构中是相互关联的。杂货铺也利用层次逻辑来布置货物。（如果要找草莓酱，到杂货店的食品区域，在摆放果酱的货架上，你能够找到不同类型的草莓酱，在同一个区域你还能找到花生酱和蜂蜜。）以层级结构的方式组织音乐也有其意义所在（流派→乐队→专辑→曲目），整个布局比较清晰，也容易理解。

如果以明显的、常规的层级结构组织事物，人们通常就能够理解并利用这种层级结构。如果要求人们对任意事物创建并维持层级结构，那就会出现问题。

22.1 层级结构

人类很难把任意事物组织成层级结构，并且记住某个具体事物的位置。马克 · 沙特尔沃斯

（Mark Shuttleworth）[1]认为层级化文件系统是 Ubuntu 系统可用性的主要来源，他说："人们在收邮件时会存储附件，但一个小时后却不知道把附件存放在哪里了。"[2]无独有偶，在名为 Improving the Usability of the Hierarchical File System 的论文中[3]，作者加里·马斯登（Gary Marsden）也描写道：

> 所有研究过文件存储程序的人都发现，除了对最熟练的用户，文件系统对其他所有用户来说都是一个较大的障碍。

马高·塞尔兹（Margo Seltzer）在他的论文 Hierarchical File Systems are Dead 中写道：[4]

> 用户不再记得自己把文件存放在了什么地方。另外，他们更倾向于通过搜索的方式来寻找自己想要的数据。我们鼓励对此持怀疑态度的读者去问一下你那些不太懂技术的朋友，他们的文件究竟放在什么地方。即便是像你这样对技术非常熟悉的用户恐怕也写不出自己邮件的存放路径吧？那么 Quicken 文件呢？你刚才编辑的 Word 文档呢？你最近运行的程序呢？为了记住这些文件的位置，你在桌面上放置了多少文件？这些文件又有多少是相互关联的呢？

如果多个用户访问同一数据，以上情况会更加错综复杂。对于某个人有用的层级结构，其他人或许根本无法理解。即便大家都认同常规的结构形式，但也常常会在应该把某个具体项放在什么位置这一问题上产生分歧。

22.2　空间

虽然我们能够理解浅显、与直觉相符的层级结构，比如家族结构和杂货店布置货物的逻辑，但我们实在是不太擅长为其他事物构建出层级结构，通常不能以较深的层级结构组织管理自己的东西。当然，你或许把所有餐具都放进一个橱柜里，把所有厨具都放在另外一个橱柜里，但那很难称得上是层级结构。你只是按照空间位置来布置，把相互关联的东西放在了一起。你根本无法通过任何层级结构找出餐盘，你之所以能找到它，仅仅是因为知道自己把它放在了哪里。

同样，你的桌子上或许是一片狼藉（见下页图），但你却很清楚地知道自己把上周收到的信放在了桌子上的哪个位置。你能做到这一点，并不是因为用某种层级结构整理了桌面上的东西，而是仅仅因为你知道自己把信放在了哪里。

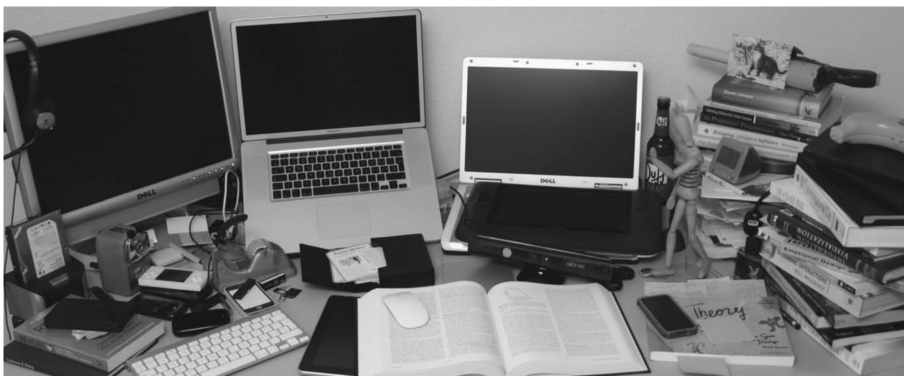
---

① 沙特尔沃斯是 Canonical 公司的前 CEO，Linux 下 Ubuntu 系统家族产品的创建者。Ubuntu 是基于 Linux 内核的操作系统，非常注重可用性。

② 关于他的文章，参见 http://www.markshuttleworth.com/archives/223。

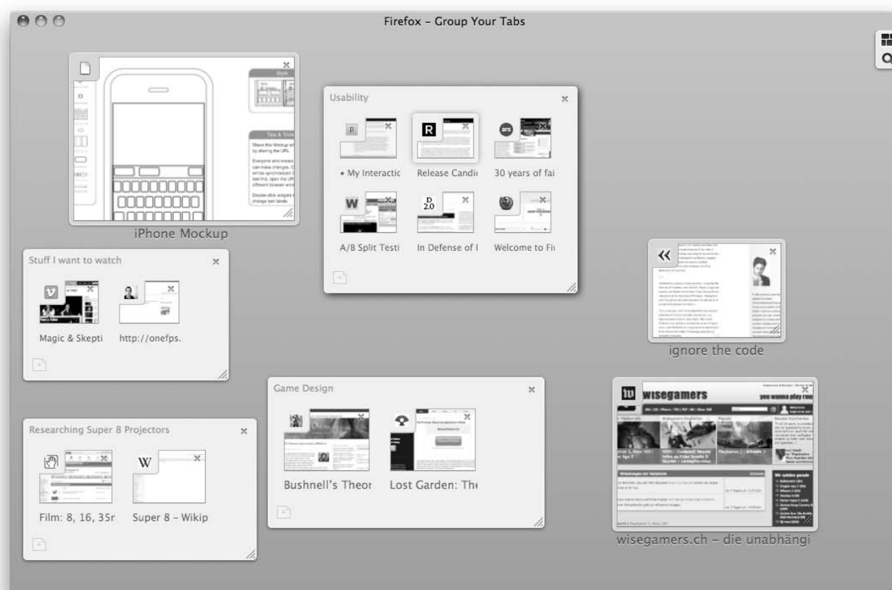
③ 该论文位于 http://people.cs.uct.ac.za/~gaz/publ.html。

④ 参见 http://www.eecs.harvard.edu/~margo/papers/hotos09/。



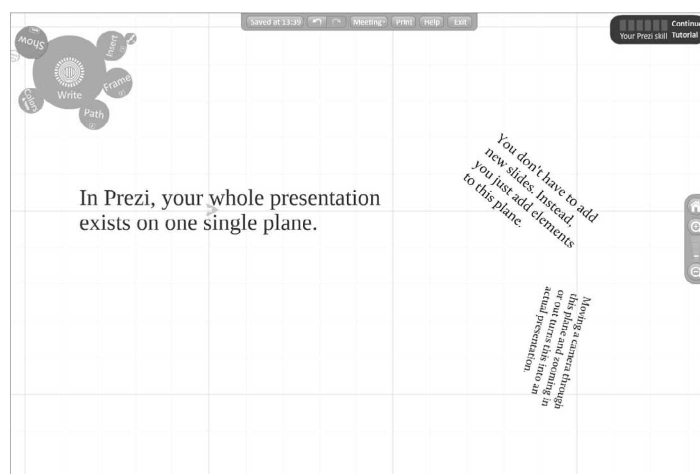
这种现象称为“空间推理”（spatial reasoning），人类已深谙此道。如果你的产品能利用人类这种对空间的感知能力，能让用户对他们的数据在空间中进行布局，那就去做吧。

Firefox 的标签分组功能就是个例子。它允许用户把所有标签以空间视图的方式放置在一个窗口中，并在二维平面上对其进行分组[①]。



另一个例子是名为 Prezi[②]的演示程序，它在平面上显示整个演示文档，通过在平面内移动摄像机镜头，可以把整个文档分解成单个的显示页面。

① 阿扎・拉斯金解释了这一功能背后的设计理念，参见 http://www.azarask.in/blog/post/designing-tab-candy/。
② 参见 http://prezi.com。



一定要记住，现实世界中的东西不会自己移动。在空间系统中，人们期望在自己设定的位置找到某个东西。一定不能背着用户改变某些东西的位置。

当用户对大量内容进行管理时，空间系统显得有些乏力。为数十项或数百项内容创建空间布局，并跟踪其每一项的位置是比较容易实现的。但如果有上千项内容，就很难做到这一点。当系统要处理成千上万的数据时，几乎无法用空间结构来对其进行管理。

22.3　时间

人类已在空间世界中生存了几千年，这一经历教会了人们如何在现实世界中组织并找到某件东西。同样，我们也变得非常善于用基于时间的方式组织管理自己周围的事物。

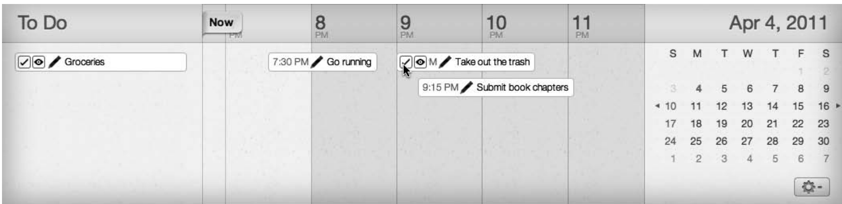


虽然你可能无法清晰地记起把上周撰写的报告放在文件夹的哪个位置了，但你可以确定确实是上周撰写了那份报告。可能也记得大约是一天中的什么时段写的，甚至还记得是在哪天写的。

因此，根据产品的不同，让用户以某种时态视图来访问数据可能会更有效。任天堂在这一点上就做得很好，他们没有采用文件系统，而是用日程表来显示以用户为中心的数据。在使用任天堂的系统时，你永远不需要对自己制作的绘图或是照片进行归档。只要记住是在哪个月制作的就行了，你还可以在月视图中看到图片的缩略图。



另一个例子是 Media Atelier 制作的 Mac 平台上的 Alarms 程序[①]，它是个日程列表，没有使用其他任务列表采用的传统层级列表形式，而是以时态视图的方式显示将要完成的事项。



如果可以，允许用户用基于时间的视图访问数据。

22.4　更好的层级结构

在某些情况下，你可能无法避免使用层级结构。如果要采用层级结构，以下这些方法可以帮助你把层级结构设计得更加可用。

① 更多内容，参见 http://www.mediaatelier.com/Alarms。

限制深度

如果只给用户一截绳头，他们很难用它把自己吊死。如果只能让用户创建较浅的层级结构，他们不太可能会迷失其中。照片分享网站 Picasa 就是这样的一个系统，它允许用户把自己的照片整理到影集中，但不能把一个影集放到另一个影集中。一个影集只能存放很多照片，别无他用。这样，用户就不太可能会丢失自己的影集，因为他们根本就不能把某个影集放到另一个影集中。

空间中的层级结构

当苹果为自己的 iOS 操作系统[①]添加文件夹功能时，它们做出了非常明智的选择：只要你不在同一个文件夹里放置超过 9 个应用程序，就能够通过文件夹的图标看到里面的程序。从技术上来说，这是用层级化的方式布局应用程序，但即便如此，用户也不太可能会找不到自己想要的东西，因为这样的层级结构看上去更像是空间布局。



允许用户在多个地方放置内容

通常各个项目并没有十分明确的归属。假如你正在使用一个文件管理系统，它包含有多个目录，如用来管理保险相关文档的 `Insurance` 目录，用来管理账单的 `Bills` 目录。如果保险公司发来一个账单，你该怎么办？相同的文件应该存在于两个不同的地方。层级化的文件系统应该允许用户把相同的文件放在多个地方。

支持标记和其他元数据

仅允许用户把项目组织成层级结构还不够。层级结构通常只能表达项目某个方面的性质。例如，对动物进行层级结构分类使用了常见的降级原则。如果你对各个生物种类之间的关系非常有兴趣，这样的分类方法会很有用。但如果要编写一个对人类有毒的动物列表，或是所有水生动物的列表，这样的分类就不能起到太大的作用。允许用户在层级结构中对各个项目的元数据进行操作，这样他们就可以组织管理那些无法通过层级结构表达的数据。

支持以其他方式访问数据

即便主要用层级结构组织内容，你也要允许用户以非层级结构的方式访问数据。比如，允许用户搜索数据。同样，如果需要追踪各个内容项的变化，应允许用户以时态视图访问层级化结构的数据。你要考虑一下用户可能会用何种方式搜索或浏览数据。

① iOS 是主要用于 iPhone、iPad 和 iPod touch 的操作系统。

BizTwit 案例

由于所有已发布的 Twitter 信息都有其发布时间，所以允许用户以时态视图来访问数据会有更大的价值，也能让用户很快找到以前的会话记录。

BizTwit 允许用户设定自动发布信息的日期。时态视图应该包括这些内容，让用户能清晰地了解到信息会在什么时间发布。图 22-1 是相关用户界面的线框图。

在这个视图下，用户可以过滤出自己想要访问的信息（已发送的信息、其他人对自己的回复、草稿、将要发送的信息和所有信息）。各个日期下的点表示这个日期内有活动，点的个数从 0（没有收到或发送任何信息）到 3（发送或接收到了很多信息）不等。后续日程下的点表示已经计划将在这一天发送信息。点击某一天可以显示当天之内的信息。这种视图对于所有使用日历应用的人来说再熟悉不过了。

这个用户界面能让用户迅速浏览自己 Twitter 账号的日程活动，很容易地找出以往的信息，也能很容易地看出将要在何时发送什么样的信息。

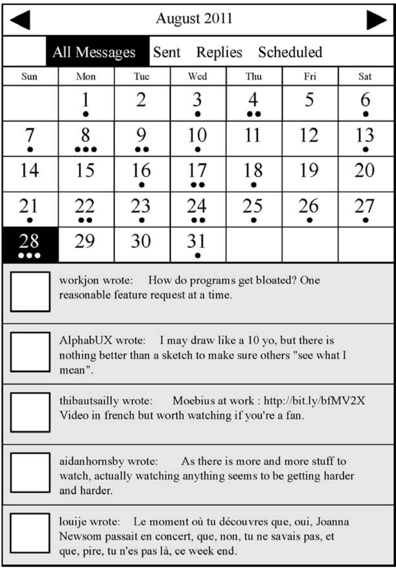


图 22-1　BizTwit 时态视图的线框图

提示

- 有时，层级结构很有效，但结构一定要清晰，能被大众用户所接受。
- 如果每个人都要自己设定层级结构，那么它的作用会降低，尤其是多个人共用一个层级结构时。

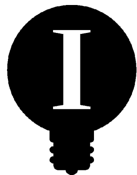
- 人类非常擅长于处理时间和空间信息。试着利用这两个因素，而不仅仅是层级结构来组织用户数据。
- 如果必须使用层级结构，应尽可能使用浅显的结构，允许同一项目出现在多个地方，支持对元数据的操作，允许用户以其他方式访问数据。

延伸阅读

皮特·莫维尔和路易斯·罗森菲尔德共同撰写的《Web 信息架构》对层级结构进行了一定程度的介绍。马高·塞尔兹和尼古拉斯·墨菲共同完成的论文 Hierarchical File Systems are Dead 也值得一读。[①]

华盛顿大学信息学院的副教授威廉·琼斯（William Jones）曾做过一个关于个人信息结构化的有趣的演讲。[②]

① 参见 http://www.eecs.harvard.edu/~margo/papers/hotos09/。
② 参见 http://www.youtube.com/watch?v=aufuHuNRqaE。



第 23 章

速　度

23

在可用性测试中，我们通常不太关注产品的速度和响应度。当用户不能理解某个按钮标签的含义时，可用性测试能很快发现问题，但它却无法暴露出产品反应过慢的问题。可用性测试中的用户使用产品的动机和实际用户是不一样的。可用性测试中的用户知道自己是在测试你的产品，即便因为必须等待产品的反应而变得沮丧时，他们也不会突然放弃使用你的产品而转向竞争产品。

但真正的用户却完全可能会这样做。技术投资者弗雷德·威尔森认为：[①]

> 速度比功能更重要。速度本身就是最重要的一项功能。如果你的应用程序慢慢吞吞，人们不会使用它。这种现象在主流用户中比在有较强能力的用户中更为常见。我认为有较强能力的用户有时会用同情的眼光看待创建快速网络应用程序所要面临的挑战，他们或许能忍受产品缓慢的反应速度。我的妻子和孩子是我观察世界的主要窗口，我在观察他们时发现，如果某些东西很慢，他们就会离开。（……）当我们看到自己投资的某些公司的应用程序运行速度变慢时，我们也能注意到公司的成长速度也不如以前那样快。现实中的观察经验证明，速度不仅仅是一项功能，也是一项用户需求。

即便可用性测试无法明显地暴露出产品在速度或是反应度方面的不足，我们也一定要关注产品的性能问题，这一点非常重要。

23.1　响应度

我们首先要关注的是，产品如何响应用户。很多研究在这方面都得出一致结论。[②]如果某一项操作能在 0.1 秒内完成，用户就认为它是瞬间完成的。如果完成时间多于 0.1 秒，少于 1 秒，用户不再认为操作是在瞬间完成的，但并不会忘记当前正在发生着什么样的事情。如果某一操作的完成时间超过 1 秒，用户很有可能会在等待的时间内走神。

在可能的情况下，最好还是保证操作能在 0.1 秒内完成。

① 关于他的演讲，参见 http://thinkvitamin.com/web-apps/fred-wilsons-10-golden-principles-of-successful-web-apps/。

② 更多信息，参见雅各布·尼尔森关于响应时间的文章，http://www.useit.com/papers/responsetime.html。

一个与此有点关联的话题是自然用户界面的持续交互问题，比如用手指拖动列表实现滚动的操作。在这些情况下，产品性能尤为重要。用户界面必须立即对操作做出反应，并且反应过程要非常流畅。用户界面一定要同步于用户的操作，并且要以较高的帧频实现这一目标。用户是觉得自己正在与实物进行交互，还是觉得只是在向电脑发出指令？这在很大程度上取决于产品的响应度。在此类用户界面中，糟糕的界面表现（如输入延迟或支离破碎的动画）将打破用户关于产品的心理模型，会极大地让人感到愤怒。（在第 13 章中，我已对这个问题进行了详细的讨论。）

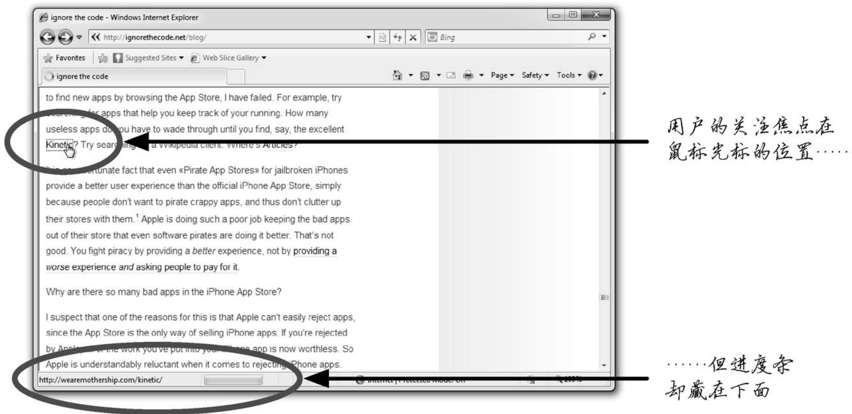
23.2　进度反馈

如果某个操作的完成时间超过了 0.1 秒，你就要提供一些反馈信息。该提供什么样的反馈取决于两点：目前进行的是什么类型的操作？要持续多长时间？

如果操作只需要 1 秒或 2 秒就能完成，把光标变成沙漏状，或添加一个小小的指示器，表明“我正在处理”就行了，这类做法不会明确告诉用户操作的处理进度。

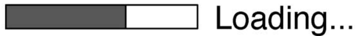


此时，我们的目的不是告诉用户操作要持续多长时间，而是要明确地告知用户，电脑已经接收到了他们发出的命令，正在进行处理。一定要“明确地告知”，这一点非常重要。观察一下人们使用网络浏览器的过程，你就会发现他们在重复点击链接，因为他们觉得第一次点击之后电脑没有接收到命令，或许你自己也是这种情况。为什么会出现这种情形呢？因为大多数浏览器显示的“正在加载”的活动图标或进度条不够明显，也不在用户点击链接时所关注的焦点位置。



没有明显的反馈告知用户，浏览器已经接收了他们的点击操作，因此用户常常不断地重复点击。

对于超过几秒钟的操作，你需要提供一些进度反馈来表示该操作将会持续多长时间。通常使用进度条解决这一问题，如下页图所示。



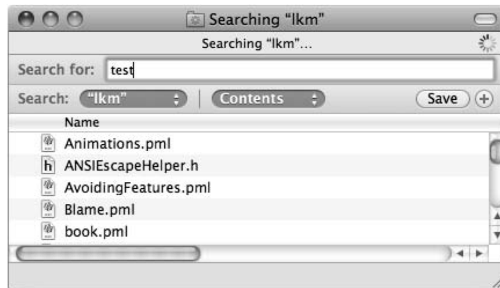
在某些情况下，或许还要告知用户电脑当前正在做什么样的事情。如果有助于用户了解自己等待的原因，那么这一点尤为重要。例如，用户从影片编辑工具中上传一个影片，该任务包括两个阶段。第一，对影片进行渲染和编码；第二，上传影片。你可以在进度条旁边添加一些简短的解释文字，甚至是一个小动画来表示任务执行状态，这样能向用户暗示当前正在执行什么样的任务。告知用户程序当前所处的状态，能向用户解释为什么这个过程如此漫长，也能让他们等待的心得到些许安慰。

如果某项任务持续时间较长，用户很有可能会把关注焦点转移到其他地方，最后甚至可能会完全忘记电脑正在执行任务。你可以用声音反馈向用户暗示任务已经完成，这样，即便用户走神，也能得知这一情况。

23.3　对速度的感知

速度本身就是用户的一种感知。用户不会用秒表来测量产品的反应时间，重要的不是操作需要持续几秒或几毫秒才能完成，而是用户对产品速度的感知。

改进速度感知的一种方法是先尽快显示部分结果。如果你正在为搜索系统设计用户界面，不要等到搜索完成时才向用户显示结果，只要找到了结果就马上显示出来。



另一种方法是确保持续时间较长的操作不会卡住产品的用户界面。如果用户可以在此过程中做其他事情，而不是傻等着操作完成，他们可能不会注意到操作持续了很长时间。

最后一种方法是在进度条上显示一些与进度成反方向运动的条纹，如下页图所示，这样会让进度看上去更快一些。[①]

① 参见 http://www.newscientist.com/blogs/nstv/2010/12/best-videos-of-2010-progress-bar-illusion.html。

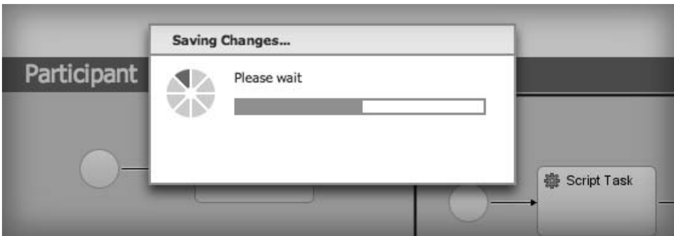


23.4　慢一点

缓慢的产品惹人讨厌，但是，非常快的产品也一样。某些时候，你的产品可能太快了。滚动操作就是个典型的例子，如果程序滚屏太快，用户会错过要找的目标。刻意让产品慢一些通常具有一定的意义，这样用户才能跟上节奏。

我在使用商务流程管理软件 Appway 时曾遇到过类似的问题。它的流程编辑器有一项保存功能，让用户手动保存对流程的更改。当人们使用这个编辑器保存更改时，通常会重复点击保存按钮，而不是只点击一次，有时甚至还抱怨保存功能不管用。

事实证明，该软件的保存过程太快了。保存用户的工作流程只需要不到 1 秒的时间。没有进度指示器用明显的反馈告知用户文档已经被保存了，但保存文档对于用户来说很重要，他们期望在点击按钮时看到某些实质性的变化。解决办法是迅速创建一个“假的”进度指示器，告诉用户保存过程已在半秒内完成了。



这样做可以让用户更加确信软件发生了某些变化。用户体验咨询师哈里・布里纳尔谈到了在数币机上发生的类似故事。[①]数币机工作速度太快了，用户甚至无法相信它已经正确地数完了硬币。为了修正类似的用户体验问题，制造商对机器做出了一些改变，以稍慢的速度显示出数币的结果，并增加了扬声器，在机器等待计数器得出最后数字时播放数币发出的声音。这样做的结果是，人们相信机器在努力地数他们的钱，更加相信数币的结果。

① 关于该文章，参见 http://www.90percentofeverything.com/2010/12/16/adding-delays-to-increase-perceived-value-does-it- work/。

提示

- ❑ 反应度和速度是产品非常重要的特征。可用性测试通常无法暴露出这两方面的问题。
- ❑ 为了让用户感觉操作是在瞬间内完成的，时间不能超过 0.1 秒。
- ❑ 对于超过 1 秒的操作，需要在用户的视觉焦点处显示一些指示，表明程序正在工作。
- ❑ 用户所感知的速度比产品的实际速度更为重要。有时候可以通过一些技巧让用户感觉产品更快一些。
- ❑ 产品的某些东西可能太快了。某些时候，人为降低产品某些功能的执行速度能改进用户体验。

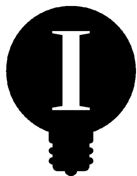
延伸阅读

雅各布・尼尔森关于反应时间的文章是关于本章话题非常好的参考资源。[①]

布鲁斯・托格纳兹尼在他撰写的《交互设计的基本原理》（*First Principles of Interaction Design*）谈到了产品的延迟问题。[②]

① 参见 http://www.useit.com/papers/responsetime.html。

② 参见 http://www.asktog.com/basics/firstPrinciples.html。



第 24 章

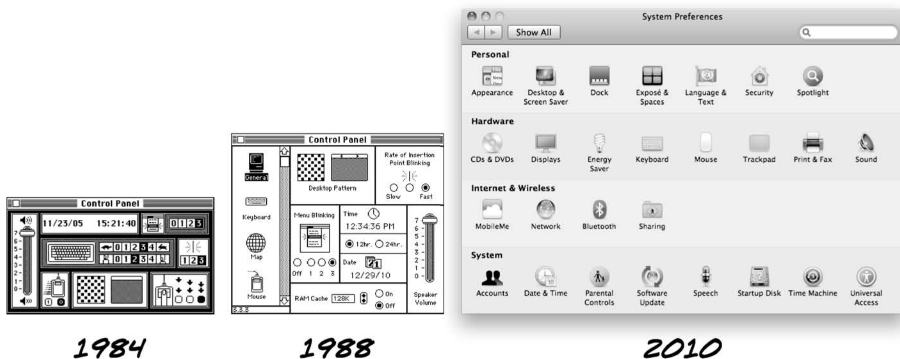
避免不断加入新功能

产品设计是一场没有终点的比赛。为了避免被其他竞争者取代，你要永不停歇地奔跑——或者，对我们来说就是：设计，设计，再设计。这场竞赛没有明显的终点，产品永远不可能做到完美无瑕。大多数情况下，我们最大的期待也只是，产品在每一次发布新版本的时候能更接近于完美一些。

就像某种鲨鱼只有不停地前进才能为自己的腮提供有氧的水，同样，如果产品长时间停滞不前就会消亡。这是个问题，因为你必须要经常为产品加入新的东西。其结果是，大部分产品都在不假思索地追求新功能。

在加入新功能之后，你的产品可能会从其初始状态（对定义明确的问题有着优雅的解决方案）迅速变得错综复杂，到处充斥着毫不相关的功能。总的来说，产品可能会解决一些问题，但大部分都解决得不是很好。如果用户无法找到解决某个特定问题所要用到的功能，那么设计一款（从理论上讲）能够解决许多问题的产品是毫无意义的。

虽然那些已了解产品的用户能够跟得上功能变化的脚步，但对于那些新用户，这样做实际上是拒他们于使用产品的大门之外，因为他们无法从简单的产品开始，随着功能的增加逐步学习如何使用产品。



更为糟糕的是，随着产品的成长，当前的用户会变成高级用户，这意味着你要继续开发更多的高级功能来满足他们的胃口。如果你这样做了，产品的客户群便会向高级用户倾斜[①]，这是个恶性循环。

你该如何管理产品的成长，避免为其加入无数新功能呢？

24.1　谨记用户的目标

考虑一下，用户想用你的产品完成什么样的目标？新功能是否能提供更好的结果？是否能让用户实现新的目标？是否能让产品更加易用？正如凯斯·塞拉说的那样[②]："人们并不关心你提供什么样的工具，只关心工具能做什么。"

24.2　五个为什么

在收到用户反馈信息时，你应该做的第一件事就是找出他们真正想要实现什么。通常，不必为产品加入新功能就能找到用户问题的解决方案。

找出问题根源的一个方法是问"五个为什么"。据说，此方法最初是由丰田公司的创始人丰田喜一郎提出的。在找出促使某些人提出特定功能要求的原因时，这一方法非常有效。你只要不停地问"为什么"（或者以另外稍微友好的方式提问），就能最终找出用户认为需要新功能的原因。不一定非要问五次，有时不足五次也会奏效，有时则可能需要问更多次才能找到最根本的原因。

让我们用一个例子说明此方法的使用过程。

> **用户**：嘿，如果你能加入一些新功能，让 BizTwit 定期为我正撰写的内容保存一份副本，那一定非常棒。
>
> **设计师**：听上去确实不错。如果允许的话，我想问你一些问题，以确保能够了解你将如何使用这一新功能。这样，我才能让它为你提供最好的服务。如果我决定要为产品加入这项功能的话，我能问问你想用这项功能来做什么吗？
>
> **用户**：当然，在撰写内容的时候，我常常需要返回到信息的早期版本。
>
> **设计师**：哦！我知道这项功能的作用了。你想要使用文档的早期版本，是有什么特定的目的吗？
>
> **用户**：我有时会更改那些不喜欢的信息。如果有早期版本的话，我就能从中复制一部分内容。

---

① 克里斯·克拉克对这一问题进行了阐述，参见 http://releasecandidateone.com/236:crotchety_old_power_users。
② 关于此人更多的文章，参见 http://headrush.typepad.com。

设计师：产品已经有了“撤销”功能了，它能让你去掉那些不想要的更改。你觉得那项功能不好用吗？

用户：我想，通常情况下用它是可以的，但当我每次保存正在撰写信息的草稿时，总有一些原因，使我无法撤销到某个状态。还有，当我试图向信息中粘贴带有格式的文本时，产品有时会崩溃，因此我已经养成了经常保存的习惯，每几分钟就存一次，这意味着“撤销”命令通常无法发挥应有的功能。

在这一例子中，用户最初的要求是为产品加入新功能。但事实证明，引发他提出要求的最根本原因在于 BizTwit 的 bug。其一，BizTwit 的“撤销”命令不能让他撤销已保存的更改；其二，BizTwit 经常崩溃。用户的问题可以通过修复这两个 bug 来解决，不需要为产品加入新功能。

看问题要更深入一点。用户真正想要的是什么？他们想用添加的新功能解决什么样的实际问题？

24.3　提升已有功能的可用性

如果大量新要求都能通过产品当前的功能集满足，那么你应该花些时间改进产品已有的功能，而不是创建新功能。同样，如果没有人能理解或使用产品的某项功能，你最好研究一下该如何改进这项功能。

实际上，你的目标不是增加全新的功能，而是让已有功能更加可用。这样做实际上是把产品变得更加简单，而不是更加复杂。同时，这样做也等同于为所有不知道产品存在这项功能或是不知道如何使用这项功能的人提供了新功能。

当然，你也可以用新功能来代替现有的功能。在考虑加入新功能时，你要考虑这样做是否会孤立现有功能。一个赞同实施新功能的观点是，虽然新功能与现有功能相同或相似，但其使用方式更好，更易用，如果是这样，你就可以移除某项现有功能。

24.4　一举多得

不要单独地加入新功能，把相似（或者看上去毫无关系的）功能当作一类问题来解决更有意义。

例如，考虑用户对某个文字处理软件的以下新要求。

用户 1：我经常写信，如果能在文档中自动插入我自己的信笺抬头就最好不过了，这是个非常有用的功能。

用户 2：我需要一种改变默认字体的方法。

用户 3：是否有一种方法能同时改变多个文档的脚注样式？

表面上看，这些要求没有什么共同点可言。然而，使用一个模板系统就能满足这些要求。为用户提供选项，让他们可以从一系列模板中创建文档，这样就能满足大部分用户的要求。

24.5 成本

每一项新功能都有其价值所在，当然我们也要因此付出代价。为了合理地评估新功能的影响，通常需要同时考虑这两方面因素。

一项功能的价值通常很明显，就是能让用户实现哪些目标。

但为产品加入新功能需要付出的代价却没有这么明显，额外的复杂性可能会让产品损失那些不想使用新功能的用户。为产品加入新功能，你就不能做其他事情，也会拖慢后续进度，因为你又多了一项必须维护的功能。如果某项功能依赖于不可控的系统，或由于其他原因易于出错，那么可能导致用户支持的成本不断增长。

除了考虑新功能所能带来的益处之外，还要考虑实施该功能所要付出的代价。

24.6 隐形的功能

如果能在增加一项有用的功能的同时不加重用户的“认知负担”，那最好不过了。隐形的功能是最佳选择，因为它们可以让产品变得更优秀，但却不增加其复杂性。你是否能改进文本渲染功能，让产品创建的文档看上去比竞争对手的产品创建的文档更好看？如果为网络应用程序增加支持 HTTPS 访问的功能，是否能直接把用户转到 HTTPS 访问并保证数据更加安全，而不让用户手动启用新功能？

这些功能都能让你的产品更加出色，但用户根本不需要知道它们的存在。

24.7 提供 API 和插件架构

没有必要自己实施所有的功能。通常，你可以让其他人介入进来提供帮助。例如，Twitter 在开始的时候只提供了极其简单的服务，但它的 API[①]可以让其他开发者介入进来，围绕 Twitter 创建完整的软件产品生态系统。Photoshop 是另一个典型的例子。虽然 Adobe 已经在 Photoshop 中添加了无数功能，几乎没有什么可添加的其他功能了，但 Adobe 仍然允许第三方开发者为其创建插件，这样可以不花自己一分钱，就能拓宽产品的市场，也没有让 Photoshop 变得更加复杂。

通过增加 API 和插件架构，你就可以避免实施那些仅对部分用户有用的功能了，而将这一工作留给其他开发者。当然，你也可以用插件架构自己创建新功能。

① API 即应用编程接口（Application Programming Interface），可以让开发者创建与你的产品进行交互的应用程序。

24.8 倾听用户的心声

一定要记住，你并不是产品的用户。最多你也只是其中一个用户，是所有用户的少数典型个例。你可能认为某个新功能非常有用，但大部分用户却持反对意见。这种情况有时被称为“内部用户问题”：设计和实施解决方案的人把类似自己的人而不是产品的实际用户作为目标用户。

你知道很多关于产品工作原理的内容，但对用户如何使用产品却不甚了解。另一方面，用户非常清楚自己将如何使用你的产品，但不清楚产品的工作原理。

通常很容易为产品增加一些无用的复杂性。你知道产品如何工作，因此它对你来说不算复杂。

让用户测试新功能永远都是明智之举。注意，人们通常不太愿意提供负面的反馈信息，因此最好给他们看简单的草绘图，而不是实际的实施过程。这样就不会让用户认为，你已经在此功能上投入了很多精力，否则他们不太愿意让你的理想破碎。让用户明确地说出负面影响也很有帮助，比如：“请说出你对这项新功能不满意的三个方面。”

你可能会对用户的反应感到惊讶不已，我就有过这种经历，一些人说我提出的新功能看上去很酷，但却找不到任何使用它的原因。

24.9 不能太听信于用户

有些人告诉微软的瑞克·斯考特，微软 Word 5.1 的 Mac 版本是该产品最完美的版本。在谈到这些人时，瑞克·斯考特撰文[①]指出，如果你问他们对文字处理软件的真正需求是什么，他们总是会要求增加至少一项特殊的新功能。他们需要的不仅仅是“Word 5.1”，而是“Word 5.1 加一项新功能”。但是所有人对新功能都有着不同的要求，如果为产品加入了所有这些新功能，瑞克说道，就相当于实现了目前最新 Word 版本的所有功能。

当用户对你的产品提出设想时，他们实际上是在憧憬一款为自己量身打造的产品，该产品提供的功能恰好是他们所需要的。而这通常会包含一两项除了他们自己几乎没人用得到的奇怪功能。

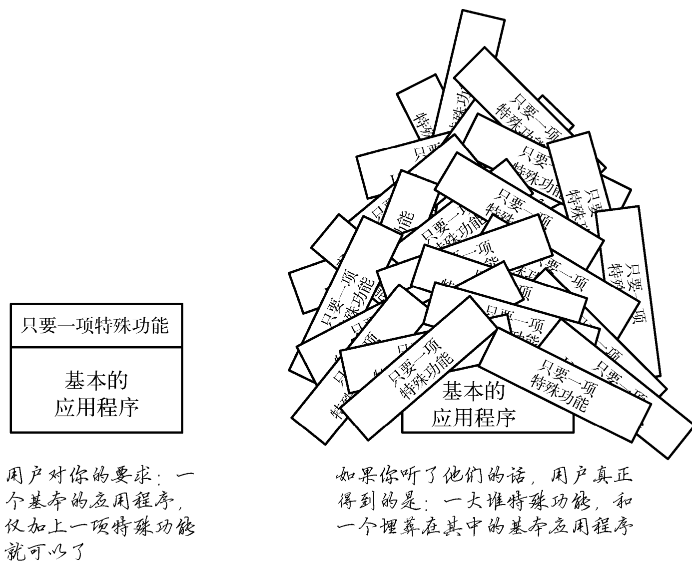
如果你满足了“我只需要一项特殊的功能”之类的要求，最终产品的大部分功能都会变成特殊功能，大众用户只使用少数功能。微软就是这样做的，但这对你的产品来说并非明智之举。

黛博拉·维安娜·托马森、瑞贝卡·W. 汉密尔顿和罗兰·T. 拉斯特最近开展的研究[②]表明，大部分用户在被问到时都说想要拥有更多功能的产品。人们认为即便目前用不到，自己也会在将来用到这些功能。但是，当这些人使用他们自己所要求的产品时，最终会被自己所要求的功能搞

① 整篇文章参见 http://blogs.msdn.com/rick_schaut/archive/2004/06/18/159325.aspx。

② 关于该研究的更多内容，参见 http://hbr.org/2006/02/defeating-feature-fatigue/ar/1。

得非常沮丧。用户认为自己喜欢有更多功能的产品，但到最后，他们却选择了功能较少的产品，甚至是某个简化的、并没有提供任何他们认为自己将会用到的功能的产品。你的产品能提供最多的功能，但这很快会变成不利因素，那些为此选择了你的产品的用户，最终会因为自己的购买行为感到痛苦不已。



同样，巴里·施瓦兹在《无从选择》中也指出，随着选择的增多，人们从选择过程中得到的满足感会相应地下降。让人们做更多的事反而会让他们不高兴。

如果你想得到满足感，让高兴的用户把你的产品推荐给他们的朋友，那么在为产品加入新功能时一定要小心谨慎。

24.10　不必让所有人都成为用户

最后一定要记住，你的产品不可能占有 100%的市场。为产品加入更多功能确实能够笼络更多的用户，但这样也会增加开发成本。如果没有提供那些具体的功能，产品对那些不具备使用能力的人将具有更大的吸引力。但是，这样做也会让产品对那些将会用到这项特定功能的人的吸引力降低。

让某些人转向你的竞争者，让竞争产品满足他们的需求吧！你的产品不可能是万能的。

艾萨克·霍尔（Dropbox 的竞争者 Syncplicity 的创始人之一）写道，他曾经冲进 Dropbox 公司 CEO 的办公室，问道："为什么 Dropbox 不支持同步多个文件夹？"霍尔说他得到的回答[1]解

① 完整的回答参见 http://www.quora.com/Dropbox/Why-is-Dropbox-more-popular-than-other-programs-with-similar-functiona。

释了为什么 Dropbox 如此流行。Dropbox 早前曾支持多个文件夹的同步功能，但在 beta 版本中对该功能进行测试时发现，这一功能会让用户产生疑惑。Dropbox 自己也无法做出正确的用户界面。该项功能实在太难用了，最后 Dropbox 放弃了它。

Dropbox 或许因为不具备这项功能而损失了部分用户，但这种“少即是多”的理念让 Dropbox 变得十分易用，最后成了同类软件中最受欢迎的产品。

让产品功能丰富是需要付出代价的。用户更热衷于那些能把一件事情做到极致的产品，而不是能马马虎虎做到所有事情的产品。

37signals 的成员在《重来：更为简单有效的商业思维》一书中说道：“如果某些用户极其热爱我们的产品，我们愿意损失一些其他用户，这是我们的准绳。”

提示

- 增加更多的功能可以让产品对那些需要这些功能的人更具吸引力，但却会让那些不使用这些功能的人对产品更没兴趣。通常，对于任何既定的功能，第二种人更多一些。
- 用“五个为什么”来评估用户的反馈，找到问题的根源。
- 不要增加新功能，而是改进现有功能。
- 添加那些能解决数个问题、能取代现有功能的功能。
- 统筹考虑实现一项新功能所需的成本和该功能所能带来的益处。
- 着重关注那些能提升产品质量却不增加用户界面复杂度的功能。
- 提供 API 和插件架构。把一些专用功能做成插件，或者留给第三方开发者。
- 在加入新功能之前进行适当的用户研究，找出用户真正想要的功能以及使用该功能的目的。
- 如果你满足了所有“我只需要这一个功能”之类的请求，最终的产品会充斥着大量不常用的功能。
- 不可能让所有人都成为你的用户，请接受这一现实。

延伸阅读

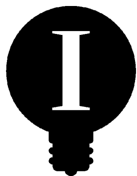
凯西・塞拉的博客上有很多关于本章主题的文章[①]。巴里・施瓦兹的《无从选择》非常值得一读。黛博拉・维安娜・托马森、瑞贝卡・W. 汉密尔顿和罗兰・T. 拉斯特共同开展的研究[②]也是不错的参考内容。

① 参见 http://headrush.typepad.com。
② 参见 http://hbr.org/2006/02/defeating-feature-fatigue/ar/1。

Mint.com 的首席设计师杰森·普托尔提（Jason Putorti）在他的博客[1]中推荐了一些测试方法，可用来决定应该实现哪些产品功能。

为何一些看似简单的功能背后隐藏着大量的复杂性？Mac 平台上流行的 RSS 软件 NetNewsWire 的开发者布伦特·西蒙斯（Brent Simmons）撰写了一些关于此问题的文章[2]。

① 参见 http://jasonputorti.com/post/229492382/how-to-avoid-feature-creep/。
② 参见 http://inessential.com/2009/07/30/anatomy_of_a_feature。



第 25 章

去掉某些功能

25

上一章我们谈到了如何避免产品功能缓慢，保证产品和用户都不会为太多的功能所累。但你可能会发现，自己的问题不再是如何避免为产品加入太多不必要的功能，而是该如何去掉产品已有的某些功能。

引起这类问题的原因是多种多样的。或许你所用的技术已经过时了，早在 10 年前，为产品加入 FTP 同步功能是绝对合理的，但现在有更好的方案来实现同步功能，FTP 就变成了产品的不利因素。或许你为产品加入了某项功能是因为后端支持它，并且是免费的，但用户反馈表明，现在已经很少有人会用它了。或许你是从其他人手中接过了某个产品，而这个人在为产品加入功能时并不像你那样精挑细选。

无论出于什么原因，现在你正处于不幸且不利的境地，必须去掉一些产品功能。告诉用户你要去掉某项功能比告诉他们你将不实现这项功能更难。产品中某些毫无意义的功能都可能是某些人的最爱。实际上，去掉任何功能都会把个别用户惹得非常不高兴。

但如果容忍那些不必要的功能存在于产品中，只能说明你对此并不在乎。这些功能很快会集腋成裘，短小精悍、受人欢迎的产品最终会变得陈旧笨重、无人问津。

本章将介绍帮助你去掉产品某些功能的方法。

25.1 研究

在决定去除掉某项功能之前，一定要确保得到足够准确的信息。你知道有多少客户会使用这项功能吗？

最好的办法是从用户处获取使用数据。你可以为产品加上“匿名发送产品使用数据”之类的复选选项，询问用户是否愿意参加此类活动。

☑ 自动发送使用数据，帮助我们改进BizTwit。

如果无法获得确切的数据，你也可以询问用户使用产品的方式。让用户参与问卷调查能找出有趣的结果。也要允许用户通过文字的形式回复调查，说明使用产品的详细过程。

Mac 和 iOS 开发者曼顿·里斯就曾通过以上方法改进了他开发的 Mac 应用程序 Wii Transfer。[1] 根据用户调查所得结果，他决定去掉产品的某项功能。他写道：[2]

> 我最终还是去掉了一项功能，对用户的调查清晰地表明该功能的使用频率不高。有趣的是，我所去掉的恰好是整个 Wii Transfer 1.0，也就是说我去掉了 1.0 版本的所有东西。
>
> 之后的两周内，我没有听到任何抱怨。我认为自己去掉了程序中让人分心、可能把用户引入歧途的东西。同时，这也能减小一部分技术支持工作，没有人会再问我关于那项功能的问题。

最后，你应该清楚地知道产品功能受欢迎的程度。为了搞清楚调查结果是否能说明某一项功能特别不受欢迎，因此要考虑去掉它，你需要回答以下问题。

- 如果只有一部分用户使用这项功能，是因为产品的设计目的就如此吗？还是说这项功能非常重要，使用率不高是因为功能本身有问题？
- 究竟是因为用户界面难以理解，还是功能被隐藏了起来，抑或是名字没取好而导致只有很少人使用这项功能？如果用户知道如何使用，该功能的使用频率会不会增加？
- 这项功能被忽略的原因是不是仅仅因为它毫无必要？

较低的使用率并不一定说明你应该去掉某项功能。例如，备份程序的大部分用户就很少使用恢复功能。很明显，这并不表明你可以去掉这项功能。

使用率低只是产品可能存在问题的指标之一。使用调查数据可以帮你得出结论，但你要根据设计知识和经验决定要去掉哪一项功能，改进哪一项功能，保留哪一项功能。

第 34 章介绍了更多关于收集数据的内容。

25.2　告知用户

在去掉某项功能之前，最好还是告诉用户你将要这样做，并询问他们的意见。因为你完全可能会在决定去掉某项功能的时候忘掉一些东西。告诉用户你将去掉哪些功能，并解释这样做的原因。

不要让用户“投票”决定该去掉哪项功能（投票结果完全无法解释为什么用户会这样选择，而且如果你决定无视投票结果，无异于鼓励用户“造反”），但一定要考虑他们的观点和所反馈的信息。正是这些人使用你的产品，因此，他们能提出关于产品使用的有效见解，而这些见解是你

① 关于他所做调查的结果，参见 http://www.manton.org/2009/07/wii_transfer_survey.html。

② 关于他的言论，参见 http://www.manton.org/2010/02/removing.html。

可能会忽略的。

25.3 提供备选方案

或许只有一部分用户会使用某项特殊的功能，但他们确实非常需要这项功能。因此，提供一些备选方案是比较好的做法。其他人创建的产品或许可以替代你去掉的那些功能，你可以联系这些人，为用户争取一些使用他们产品的折扣。

当 Bohemian Coding 公司[1]的工作人员想要去掉他们所开发的程序 DrawIt 中的位图功能时，他们联系了 Flying Meat 公司[2]的格斯・慕勒。他们达成了一项协议，所有 DrawIt 的客户都将得到一份 Flying Meat 的图像编辑器 Acorn 的免费副本。Bohemian Coding 的皮耶特・奥姆威利告诉我：

> 我收到了各种各样的用户反馈信息。我在更新产品时改进了矢量编辑功能，一些人对此非常高兴，说他们从来就没有用到过位图编辑功能。另一些人对此抱怨不已，因为他们只使用位图编辑功能，但我现在可以为他们提供 Acorn。最后，我还听到了其他一些人的抱怨，这些人非常喜欢既有矢量图编辑又有位图编辑功能的软件。幸好，这种人非常少。总体来讲，这样做还是不错的，最应该感谢的还是格斯・慕勒的无私奉献。

如果你找不到类似的解决办法，那么就为将要去掉的功能独立开发一款较小的附加产品。但是这样会让用户感觉到，你至少会在基础层面为该应用程序提供技术支持。如果你并不打算负这样的责任，最好还是不要这样做。

如果以上方法都不可行，你还可以将产品的旧版本维持一段时间。这样，那些仅使用你所要去掉功能的人就还有机会下载旧版本。如果需要，你还可以在自己的电脑上进行备份，以备后用。

25.4 开发产品的人是你

一定要记住，对产品负责的人是你自己，这非常重要。如果你的产品不再适用于用户，他们可以转向其他产品，而你却不能这样做。用户不知道某项功能的受欢迎程度，但你了解这一点。用户不知道维持某项功能，为其提供支持的成本是多少，但你很清楚。

受制于产品的人是你，因此要确保它仍然是你值得付出、引以为傲的东西。

提示

- 有时候，你应该去掉产品某些已有的功能。这从来都不是一个轻松的决定。

① 参见 http://www.bohemiancoding.com。
② 参见 http://flyingmeat.com。

- 收集使用数据，找出用户使用产品的方式。
- 如果决定要去掉某项功能，在执行决策之前从用户处取得一些反馈信息。
- 提供一些备选方案。

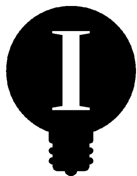
延伸阅读

Mac 的开发者布伦特・西蒙斯曾写过一些关于移除产品某些功能的文章。[①]

产品经理杰夫・莱斯在他的博客中提道，不要害怕去掉产品的某些功能。[②]

① 关于此主题有一篇非常好的文章，参见 http://inessential.com/2008/07/22/more_about_deleting_features。

② 关于他的文章，参见 http://www.goodproductmanager.com/2008/02/17/do-not-be-afraid-to-remove-features/。



第 26 章

向电子游戏学习

26

我们想让自己的产品充满乐趣，但如何才能做到这一点呢？有趣的东西很多，比如，某些人认为处于紧张状态能带来乐趣，他们喜欢过山车，乐于在鬼屋中探险。但在大多数情况下，我们不能采用这种“越恐怖，越有趣”的策略。实际上，我们在使用程序、访问网站时体验到的普通的乐趣与心理学教授米哈里·契克森米哈赖所指的“心流”有一定关联。

26.1　乐趣是什么

幸好，现在已做过很多关于这种乐趣的心理研究，并且研究结果都还比较统一。在其著作《幸福的真意》中，米哈里·契克森米哈赖说道，当达成以下条件时，人们会体验到乐趣：有自己所追求的目标；有一种衡量目标实现程度的方法；有对成功的持续反馈；技能足以应付挑战，不会超出目标太远，也不会达不到。他还描述道：“目前为止，最大程度的愉悦体验来自于一系列由目标导向、由规则约束的活动，这些活动要求人们投入精力，没有足够的能力，无法完成。”

同样，拉夫·考斯特在他的著作《游戏设计的快乐之道》（*A Theory of Fun for Game Design*）中给出的结论是，那些玩电子游戏的人在能完成任务时会体验到乐趣。游戏呈现给玩家的是看似困难，但值得在游戏世界中为之努力的难题。玩家所具备的解决这些问题的能力正是游戏的乐趣所在。他写道：

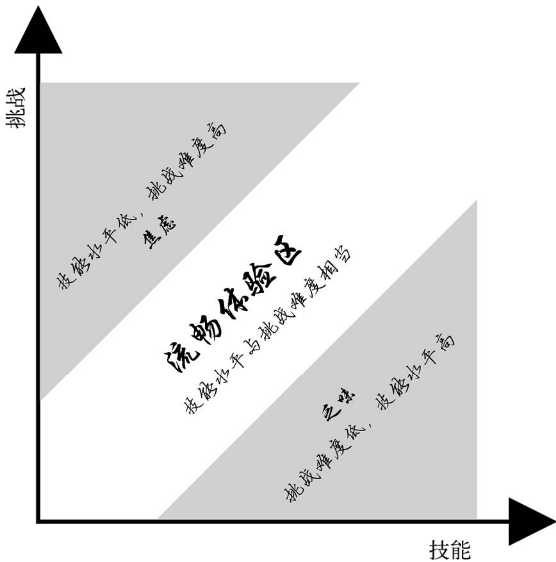
> 乐趣是大脑经历的所有美好体验——内啡肽会释放到我们的大脑系统中。（……）这种化学物恰好在我们学会了某些东西或完成了某项任务时微妙地释放出来，这一释放过程总能让我们开始微笑。毕竟，它对于我们学习的东西非常重要——因此，我们的身体回报以快乐。在游戏中找到乐趣的方式有很多，但这是最重要的一种。

让我们快速重复一下要点。当人们经历到以下三种情况时会体验到乐趣。

- 面临富有意义的挑战或任务。
- 有一种方法来衡量自己是否更接近于完成挑战。
- 拥有克服挑战的能力。

作为真正的挑战，任务不能让用户感觉太简单，那样会让他们觉得很无聊；同时，任务也不

能太难，否则用户会感到很沮丧。只有当技能水平和挑战相匹配的时候，人们才能体验到乐趣。



电子游戏产生乐趣的原理与应用程序或网站产生乐趣的原理非常相似。

26.2　产品与游戏的区别

目前我们讨论的内容既适用于游戏，也适用于你的产品。但是，它们有两个不同之处：任务来源不同，设定任务难度的人也不同。

游戏要提供任务，但产品不需要

在玩游戏时，游戏本身要提供任务。但对于你的产品而言，任务来自于用户。

我有时会听到用户界面设计师这样说：

> 游戏让玩家完成一些不必要的任务，比如收集硬币。但人们玩游戏时能体验到乐趣。因此，我的应用程序让用户完成一些不必要的任务也应该是没问题的，用户依然会在使用产品时体验到乐趣。

这种似是而非的理论为那些过分复杂的设计提供了借口。那些设计提供了太多的事情让用户去做。游戏也给玩家提供了任务：邪恶的鲍泽绑架了公主，玩家要骑着海龟去营救公主。

但是，你的产品不是游戏。你不需要给出任何任务，用户会自己找任务，比如，创建一个影片或一个演示文档，或找到某个特定的信息片段。应用程序或网站正是用来完成这些任务的工具。

游戏需要控制难度，产品不需要

对于游戏来说，设计者要控制任务的难度。对于你的产品而言，任务的难度由用户控制。

这种差异看上去有点儿微不足道，但却影响着产品的设计过程。我曾听到某些设计师这样谈论难度：

> 有些游戏比较难，但人们仍然乐此不疲。因此，我的应用程序难用一点儿应该问题不大，这样正好能为用户带来更多的乐趣。

如果困难与实际任务毫不相关，那么一款难用的应用程序或网站并不能呈现出有意义的挑战，只能让用户感到自己很傻，无法完全掌握使用产品所应具备的技能。

你不需要把任务设计得更难。如果用户感觉到自己要完成的任务易如反掌，他自己会提高要求的。例如，为自己的影片加入更复杂的过渡和标题，为信件制作漂亮的抬头，在编辑照片时学习如何使用图层效果等。

通过模仿游戏把产品变得更难用还会引发另外的问题。研究表明，现在的大部分游戏实际上都不难，只是被特别地、精心地设计得看上去很难而已。

在任天堂具有创意的游戏《超级银河战士》的开头处，女主角被巨大的怪物攻击，玩家几乎无法进行战斗，需要使出浑身解数才能一边躲闪怪物的攻击，一边发起反击。当能量快耗尽、她以为自己要挂了时，怪物却抱头逃窜了。哇，太难了！稍微逊色一点的玩家根本做不到这一点！

实际上，游戏中的怪物总是会在玩家即将要挂掉的时候投降，任何人都能做到这一点。但游戏给人的感觉是一个几乎无法超越的挑战。

游戏开发师克里斯·普鲁特在他的文章 Hot Failure:Tuning Gameplay With Simple Player Metrics[①]中介绍了一个系统，用来调节 Android 游戏 Replica Island[②]的难度。为了对游戏进行测试，他向玩家发布了能报告游戏角色死亡位置的系统版本，并根据测试所得的数据创建了一幅热成像图，如图 26-1 所示。该图显示出了每一难度级别下游戏角色死亡次数最多的位置。颜色越亮，表明玩家在此处失败的次数越多。

通过热成像图，他找出了难度过高的区域和玩家因受到不公平待遇（比如玩家所控制角色在跳跃时看不到陷阱）而失败的区域。

另外，他还为游戏加入了一个动态调整难度的系统，为那些经常失败的玩家降低难度。他解释道：

> 在玩家连续经历一定次数的失败之后，这个系统会悄悄地增加玩家角色的生命值，并提升攻击力。

① 参见 http://www.gamasutra.com/view/feature/6155/hot_failure_tuning_gameplay_with_.php。

② 参见 http://replicaisland.net。

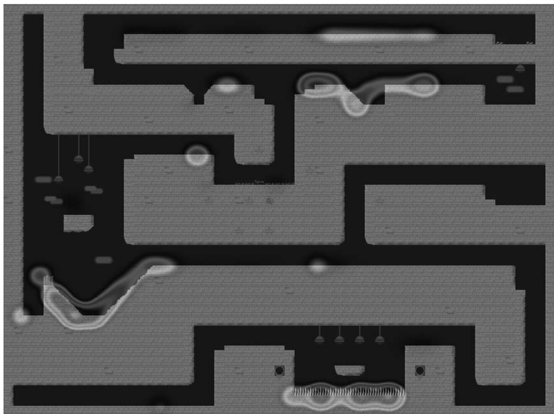


图 26-1　显示 Replica Island 中游戏角色死亡位置的热成像图

现在的大部分游戏都使用了类似的系统。Sucker Punch 游戏工作室开发出了诸如 *inFamous*（《恶名昭彰》）和 *Sly Cooper*（《怪盗史库柏》）系列之类的游戏，布鲁斯 · 奥博格是该工作室的创始人之一，当我和他交流时，他说：[①]

> 我们想让人们以稳健的步伐在游戏中前进，不想让游戏角色经常失败。如果发现所有玩家都在某个地方失败，玩家所控制的游戏角色都在某个位置死亡，那么可能就要对那个位置的内容进行修改。(……)
>
> 基本来说，无论玩家的技术有多好或多差，我们都想让每个人在玩游戏的时候高兴，想让每个玩家都能推进游戏剧情向前发展，并以此获得乐趣。

现在的游戏都很没意思，因为太难了。有趣的游戏之所以有趣是因为开发者进行了大量的测试，让游戏看上去具有挑战性，但保证玩家能克服所有遇到的难题。经常失败并不能带来快乐，但克服看上去很难的挑战却让人很快乐。

你可以将这一策略应用于产品。人们真的很喜欢 iPad 上诸如 Garage Band 之类的应用，因为它包含了一些比较难（创建优美的声音）但实际上却能让人很好地完成的事情。这恰恰是在游戏中发现的设计理念：把挑战设计得看上去很难，同时保证人们能够完成它。

游戏能教给我们的东西还有很多。

26.3　我们能从游戏中学到什么

对于如何把事情变得有趣，游戏设计师已有数十年的经验。如果能时刻记住游戏与产品的差异所在，用户界面设计师就能从游戏设计过程中学到很多东西。

① 在 http://wisegamers.ch/artikel/417/gamescom_interview_bruce_oberg_infamous_2/可以看到整个访谈过程，页面的下方有采访内容的英文版文字记录。

进度

托马斯·W.马龙在他的论文 Heuristics for Designing Enjoyable User Interfaces: Lessons from Computer Games[①]中提道：

> 无论是玩玩具还是使用工具，用户都需要一些反馈来了解目标已实现的程度。在游戏中，这种反馈由游戏中的一些东西来提供，比如游戏 Breakout 中未击中的砖块，游戏 Darts 中未正确击中数字线的飞镖的位置。一定也有相似的办法把外部任务的反馈信息融入到工具中去。例如，贝尔实验室开发的 Writer's Workbench 文本分析程序可以衡量手稿的各种风格特征，如词汇长度、句子长度、使用主动语态的句子的百分比等。这些针对撰写可读手稿的外部目标的基本性能反馈，可能会加强用户使用工具所面临的挑战。

换句话说，度量某件事情的进度并提供反馈信息，能让一项活动更加有趣。

诸如 Capcom 的 Ghosts'n Goblins 之类的游戏向玩家显示了地图概况。玩家每打通一个关卡之后，游戏中的骑士就前进一段距离。这显示出了玩家的任务，并提供了即时的进度反馈信息。EpicWin[②]是 iPhone 上的一个任务列表程序，它采用了相同的隐喻手法，告知用户列表上任务的完成进度。



Ghosts'n Goblins 中进度的表示



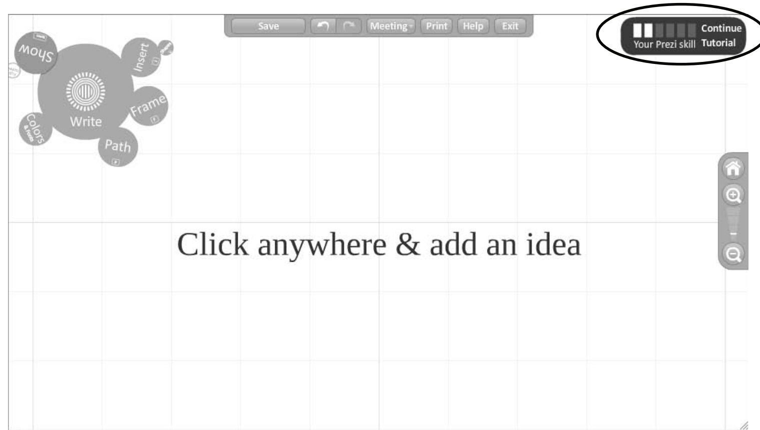
EpidWin 中进度的表示

同样，在线演示文档程序 Prezi[③]显示了一个小的进度条（见下页图），告知用户已经学习过了多少教程。在完成教程之前，无论用户在做什么，这个小图标一直可见，提醒用户当前的学习进度。

① 参见 http://portal.acm.org/citation.cfm?id=801756。

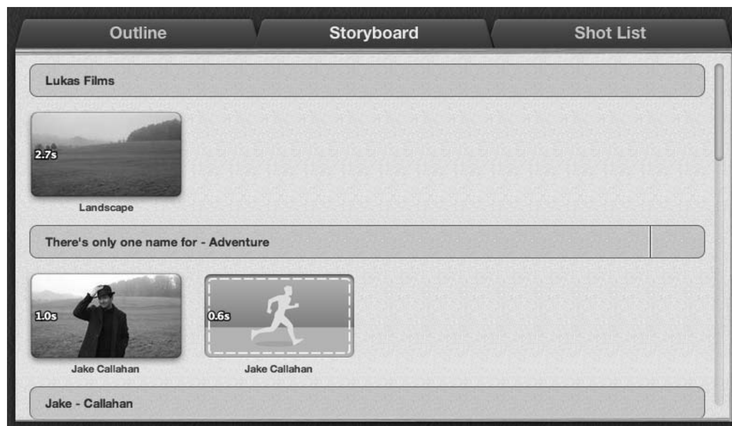
② 参见 http://www.rexbox.co.uk/epicwin。

③ 参见 http://prezi.com。



我通常都不学习程序的教程，但这个简单的进度条诱使我学完了 Prezi 的教程内容。

对进度的表示不一定非要这么明显。在 iMovie 中创建一个电影预告片时，该软件用用户已创建的场景序列表示已完成的进度。



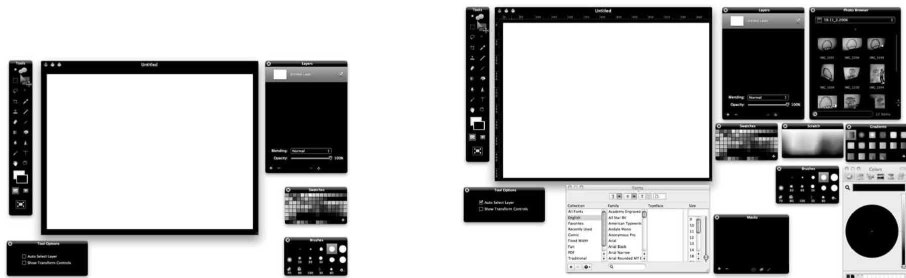
这一做法也能告知用户当前正在执行的任务及其进度。

技能成长

游戏设计中的技能成长或学习系统是另一个可用于产品设计的元素。游戏会随着玩家技能的成长变得更难。如果能做到的话，产品的用户会向更难的问题发出挑战。把用户保持在“流畅区域”之内，挑战的难度应该与用户的技能相匹配，产品需要随着用户技能的成长而成长。

图像编辑软件 Pixelmator[①]就是个很好的例子（见下页图）。最初，它的界面看上去很简单。但随着自身技能的成长，用户会发现并学习到那些最初没有注意到的新东西。

① 参见 http://www.pixelmator.com。



第一眼看上去，Pixelmator貌似很简单…… ……但软件的难度会随着用户技能的提升而加大

为了让用户提升技能，产品要有一定的深度。既要便于新用户学习，但还要为高级用户提供高级功能，让他们进行探索。托马斯·W.马龙建议把这一过程明确设计成产品的一部分：

> 例如，我们可以对多层文本编辑器进行设计，让初级用户只需要知道少数简单的命令就可以使用，高级的用户则可以使用那些更复杂、更强大的功能。
>
> 关键在于，多层系统不仅有助于解决简单易用和强大功能之间的权衡关系，还能改进使用系统所要面临的挑战。用户能够从陆续掌握更多的高级技能中获得自信与乐趣，如果把层设计成系统的一部分，用户就能更频繁地获得这种乐趣。

探索与奖励

现在的游戏通常把游戏进程与探索结合在一起，用所谓的“成就”（achievements）传递进度信息。微软的 Xbox 第一次引入了“成就”这一概念，之后，其他游戏平台纷纷效仿。有些系统称之为战利品、挑战、奖章等等名称。目前，很多常规应用程序和网站都已经开始效仿。

成就是系统给予用户的奖励，原因是用户发现了产品的新功能或做出了某些正确的操作。成就用来鼓励用户进行试验，也用来奖励正确的操作行为。

手机版的 Audible 用奖章和使用统计数据来表示用户使用产品的程度，鼓励用户听更多的有声读物，如图 26-2 所示。Audible 还用这些数据对用户进行不同层级的划分，包括从“新手”到“大师”的不同称谓，鼓励用户更多地使用该程序。

Foursquare 是个基于地理位置信息的社交网站，它利用秘密奖章鼓励用户进行探索：用户使用 Foursquare 找出罕见的地点会得到奖励。

问答网站 Stack Overflow[①]用奖章鼓励用户良好的行为。除此之外，Stack Overflow 还奖给用户一些点数。它甚至还提供了名为“用户鉴别力”的 HTML 片段，用户可以将其嵌入到自己的网站中来显示自己所得的点数和奖章。

① 参见 http://stackoverflow.com。



图 26-2　Audible 中的使用程度反馈

竞争

公开展示点数引入了游戏中另外一种常见元素：竞争。Nike[①]和 RunKeeper[②]之类的系统用成绩和比赛来吸引用户更多地使用它们的系统。

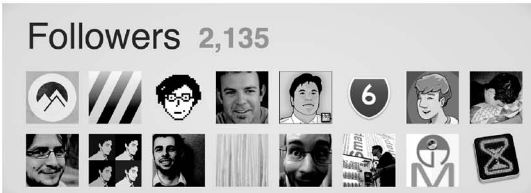
Twitter 是另一个用竞争鼓励用户的系统。在线社区架构专家金乔爱（音译，Amy Jo Kim）认为[③]，Twitter 中的粉丝不像是一个典型的游戏成绩，更像是一个收集物品的比赛。

收集物品

在 Twitter 中吸引更多的粉丝非常像收集棒球卡或玩 Pokémon 时收集怪物。



收集一个新的Pokémon……



……有点像在Twitter中吸引到一个新粉丝

很多游戏和应用程序中都有收集物品的元素。比如，iPhone 用户就有收集应用的倾向，其部

① 参见 http://nikeplus.com。
② 参见 http://runkeeper.com。
③ 她发表的关于此主题的演讲参见 http://www.youtube.com/watch?v=ihUt-163gZI。

分原因在于漂亮的图标看上去很有收集价值。用户感到收集这些图标很有趣，即便他们并不使用自己收集的大部分应用。

一致性规则

好游戏非常容易理解，拥有一套固定而一致的规则。如果游戏中的规则发生了变化，玩家就会感到自己被欺骗了。你的产品也是如此。比如， 不管在什么地方，不管应用于什么对象，同一个工具应该总是能实现相同的效果。凯特·萨兰和埃里克·兹莫曼在《游戏规则：游戏设计基础》（*Rules of Play: Game Design Fundamentals*）中谈到了电子游戏中规则的作用，他们写道：

> 规则应该是完整且非常明确的。如果你想要玩一款棋类游戏，但当你在某个地方落子后将会发生什么事情是不明确的，那么必须清除掉这种模棱两可的情况游戏才能继续进行。（……）
>
> 规则可以在不同游戏间重复，也适用于不同的玩家。

如果主宰产品的规则是模棱两可或不可重复的，那么用户将无法形成关于产品工作原理的正确心理模型。

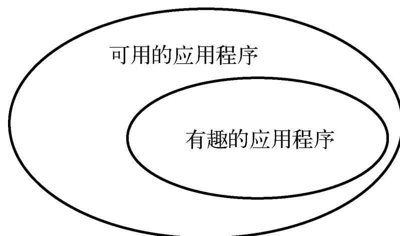
这些在游戏中常见的元素能让产品的使用更有乐趣，同时也能吸引用户对产品进行探索，学习更多的技能，执行正确操作。

26.4 趣味性与可用性

可用性有点像可食用性。[1]很多东西都可以食用，但仅仅具备可食用性并不能保证某个东西是美味可口的。同样，很多东西都是可用的，但这并不能保证我们有使用的欲望。

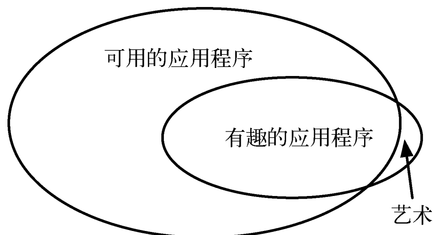
虽然可食用不是美味可口的充分条件，但它是必要条件。同样，可用性是用户获得乐趣的必要不充分条件。

因此，虽然不是所有可用的产品都能够让用户获得乐趣，但所有有趣的产品必须具备可用性——否则，它只能让用户感到沮丧。



① 并不是我想出的对比可用性与可食用性之间的差异。我在亚伦·华尔特的文章 Emotional Interface Design: The Gateway to Passionate Users 中看到了这种提法。关于该文章，参见 http://thinkvitamin.com/design/emotional-interface-design-the-gateway-to-passionate-users/。

实际上，这一结论并不完全正确。有一种产品虽然没有用处，但一样能为用户带来乐趣：艺术品。



如果征服用户是你的唯一目标，那么你实际上是在创作艺术品。本书并不介绍与此相关的内容，而是介绍人类用来达到某个具体目标的应用程序和网站。

维克多·帕潘尼克在其著作《为现实世界设计》（*Design for the Real World*）中非常严厉地批评道："创作者们不惜牺牲观赏者和消费者的钱袋子，以自己为中心，毫无忌惮地表现自我，这一现象如癌症般在艺术领域扩散，又蔓延到手工艺领域，最后连设计界也难逃厄运。"他为那些不再把消费者的需求铭记在心的设计师们扼腕叹息。

让产品充满乐趣是非常不错的目标，但永远不能以失去可用性为代价。

提示

- ❑ 仅仅拥有可用性并不能保证产品使用的趣味性。但是，如果产品不具备可用性，就无法带来趣味性。
- ❑ 如果拥有一个有意义的挑战目标、一种衡量这一目标完成程度的方法、克服这一挑战的技能，人们就会体验到乐趣。
- ❑ 不要提供挑战，用户会自行寻找挑战。
- ❑ 你不一定要设定难度。用户会自己寻找适合的难度来挑战自己。
- ❑ 为用户进度提供反馈，并允许用户随着克服挑战而成长，这能为用户带来更多的乐趣。
- ❑ 找到一种方法，奖励用户对产品的探索和正确的操作行为。

延伸阅读

米哈里·契克森米哈赖的《幸福的真意》和拉尔夫·考斯特的《游戏设计快乐之道》都深入介绍了什么样的东西能带来乐趣。凯特·萨兰和埃里克·兹莫曼合著的《游戏规则：游戏设计基础》以及托马斯·W.马龙的论文 Heuristics for Designing Enjoyable User Interfaces[①]也值得一读。

① 参见 http://portal.acm.org/citation.cfm?id=801756。

Replica Island 的开发者克里斯·普鲁特撰写了一篇非常好的关于游戏设计的博文[①]，其中谈到了可用性的问题。

在 YouTube 上，Shufflebrain 的 CEO 金乔爱发表了名为 Putting the Fun in Functional: Applying Game Mechanics to Functional Software 的演讲[②]。

① 参见 http://replicaisland.blogspot.com。
② 参见 http://www.youtube.com/watch?v=ihUt-163gZI。

Part 3

第三部分

实 施

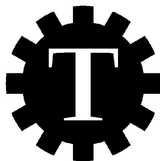
终于可以开始编写代码了！

我们已经完成了用户研究，发现了用户存在的问题，也找到了解决方案；已经提出了设计方案，并进行了测试，还迭代执行了这一过程，使尽了浑身解数使得设计方案更有用，更可用。

此时，你已经知道产品大概是什么样子了，那么就要开始创造它了。

但这并不是设计流程的结尾。实际上，设计过程才刚刚开始而已。这是实际检测设计方案有效性的机会。如果发现了无效的东西，就要尽早地改变产品的设计方向。

一旦产品问世，你将开始创建下一版本。但你如何知道该保留哪些东西，该改进哪些东西呢？诸如 A/B 测试和收集使用数据之类的技术将帮助你制订这些决策。



第 27 章

游击队式的可用性测试



什么是游击队式的可用性测试？

可用性测试通常都需要邀请别人到你跟前。这带来了很多问题，比如日程安排、建立简单的测试实验室、寻找测试者帮助你进行测试等。一件简单的事情就能规避这些问题：到用户那里去，而不是让他们到你这里来。

为什么要这样做？

尽管你会竭尽所能来规划并设计产品，但在真正的用户参与测试之前，你仍然不知道产品究竟有多好。[①]走出工作室，寻找一些人来对产品进行简单的测试，这样你就能很快找出设计方案的哪些内容需要改进，就能在很容易改变代码的情况下修复这些问题。如果现在不花点时间找出产品的问题，那么问题最终会在产品发布之后暴露出来。

为什么叫“游击队式”的测试？

我们称这种测试为游击队式的可用性测试，表面上看，是因为它与游击式战法相似，规避了传统可用性测试的很多工具和技术，支持更为简单、不需要很多人参与的测试方法，特别依赖于流动性和让用户感到惊喜的测试成分。然而，我个人认为，它之所以被称为游戏队式的可用性测试，主要是因为我们这些程序员和设计师想让自己从事的工作听上去比实际情况更粗野蛮横。嗨，看这里，代码忍者和明星设计师！

① 专业的观点是，测试是无法取代的，参见 http://uxmyths.com/post/3086989914/myth-30-if-you-are-expert-you-dont-need-to-test-your-des。

是否存在前提条件？

存在。你应该已经开始编写代码，至少有一部分用户界面应该已经可以工作。你还要保证产品能在便携设备上运行，如笔记本、手机或 iPad 等。

游击队式的测试风格

在典型的可用性测试中，你需要邀请人们到你所在的地方。在精心准备的工作间内，你坐在他们面前，让他们执行特定的任务，然后记录他们的行为。然后你就可以利用在这个交互过程中得到的信息来改进用户界面的设计。

执行这样的测试需要做一些准备工作。你需要搭建一个测试场地，想出一个记录测试者行为的方法，寻找测试者，预约测试时间，或许还要向他们付参与测试的费用。

可用性测试已经在软件社区内变得非常普遍，人们已想出了一系列技巧来规避所有的流程问题。这通常被称为游击队式的可用性测试。

它与常规可用性测试的最大区别在于，我们到测试者所在的地方去，而不是让他们到我们所在的地方来。

27.1 测试频率

游击队式的测试很简单，几乎不需要做任何准备，可以随时开展。因此，当你需要别人回答某个问题时，游击队式的测试通常是最好的选择。

比如，你已经想出了一个新的办法，让人们在你的 Twitter 应用中发送信息。你实施了设计方案，方案也是有效的。但人们会用它吗？不要自己猜测，要通过测试来寻找答案。

27.2 测试的准备工作

你需要在便携设备上运行产品代码。如果你的产品依赖于某项服务，或者产品本身就是一项服务，那就准备一个来宾登录账号，这样测试者就不必再提供他们的邮箱地址或其他个人信息了。如果产品需要有 Internet 连接，确保测试的地方可以上网。检查便携设备电池的电量是否是满的（如果可能且必要的话，最好准备一个备用电池）。

你还要考虑该如何介绍产品，并设计出测试者需要执行的简单任务。然后确保产品在执行这些任务的过程中不会出现 bug，运行每一项任务，看看产品是否有效。你的目标是找出用户界面的问题，而不是代码中的 bug。

如果需要，你还可以在电脑上安装诸如Silverback[1]或Jing[2]之类的软件来记录用户的行为。

① http://silverbackapp.com。
② http://www.techsmith.com。

27.3　如何寻找测试者

只要去附近的咖啡厅逛一逛，寻找那些没有在读报纸或摆弄笔记本电脑的人，然后询问他们是否有兴趣参与一个简短的可用性测试。

如果你在写字楼里上班，应该可以在饮水机附近找到人，也可以找那些正在抽烟的人。除非你想要专门测试现有用户对于产品新功能的看法，否则尽量不要找那些对产品非常熟悉的人来进行测试。

27.4　测试者的数量

由于在游击队式的可用性测试中你不必安排测试者，因此，你可以请很多人来参与测试，直到你认为已经得到了问题确定的答案为止。三到五人通常是最好的，但即便只有一个测试者，常常也能发现设计方案潜在的问题。

27.5　执行测试

这一步非常简单。寻找一些愿意花 5 分钟进行测试的人，向他们解释你究竟要做什么，应用程序是用来做什么的，你想让他们执行什么样的任务。如果要记录整个测试过程，你要征得用户的同意。

> 我们正在创建一款新的 Twitter 应用。你知道 Twitter 是什么吧？OK，很好。我们正在设计这款应用，为了确保设计方案的正确性，我们想看看人们是否知道如何使用这款产品，因此，我们要测试的是设计方案，而不是你。如果你不能立刻完成某项任务，也不要因此而担心，因为这正是我们要寻找的反馈结果。它表明我们的设计方案存在一些错误。如果你允许，我们将记录这次测试，这样以后就可以返回来查看整个测试过程。我们保证不会公布所记录的内容及其他类似的东西。OK？好的。对于第一项测试，我们想测试“新信息”的用户界面。我们已经登录进了该应用，只是想看看用户是否知道该如何发布 Twitter 消息。在测试过程中，你可以随意说话，也可以说明使用该应用的思维过程。

然后，坐在一边观察测试者并记下笔记。

测试者很可能马上就遇到了问题。如果觉得他很沮丧，你可以介入其中给予提示或暂停测试，否则就一直保持沉默。

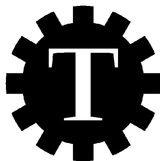
我会在后续章节中谈到更多关于如何执行可用性测试的内容，但在目前阶段，你只需要知道以上内容就可以了。

27.6 测试结果

测试一结束，你就能发现一系列问题。可能有好几个人都遇到了相同的问题，那么你要首先关注这些问题。考虑一下为什么用户不认同你的设计以及如何修正设计。然后实施修正方案，再次进行测试，直到用户接受了设计方案为止。

提示

- 可用性测试可以非常简单，但一样能产生有用的、可行的结果。
- 如果要对用户界面的变更进行测试，你可以执行非正式的、游击队式的可用性测试。
- 测试的准备工作包括：把产品安装到便携设备上，保证产品可以运行，设计出一些简单的测试方案。
- 到附近的咖啡馆找一些人，或者在你所工作的办公大楼里找那些有空闲时间的人，询问他们是否愿意花 5 分钟帮助你进行测试。
- 如果找到了测试者，向他解释测试过程和产品的前期知识，并给他一项简单的任务。
- 不要打扰测试者，除非你觉得测试者正变得沮丧或卡住了。做测试记录。
- 在进行数次测试之后，你会找出用户界面仍然存在的问题。修正这些问题，再次进行测试。



第 28 章 可用性测试



什么是可用性测试？

在关于纸质原型的章节中（第 11 章），你已经初步了解了可用性测试。现在，我们将用可运行的代码执行同样的测试。在很多方面，对代码的测试实际上要比对纸质原型的测试更容易，因为你不需要再模拟电脑。对可运行的应用程序进行测试也能赋予你更多执行测试的自由。本章将回顾前面介绍的一些概念，同时介绍一些纸质原型测试之外的新概念。

本章主要介绍可用性测试及其准备工作，后续两章将介绍可用性测试的执行过程。

本章介绍的有些概念对于较大的产品开发团队很有效，有些则适用于较小的团队，甚至是那些想自己执行整个可用性测试的个人。即便不是所有内容都能直接应用于你的实际情况，但总有一些内容适用于你。

为什么要这样做？

如果你想知道自己的设计是否有效，就要看看其他人是否能理解它。利用前一章介绍的简单的游击队式可用性测试能找到这一问题的答案，但执行更为深入的可用性测试能得到更好的结果。

是否存在前提条件？

存在。你要有可运行的代码，而且用户界面至少有一部分应该已经可以工作。

你还应该阅读第 11 章，那一章介绍了此处和后续章节都会用到的一些概念，后面不再赘述这些内容。

28.1 可用性测试的成本并不一定很高

可用性测试的基本目标是观察用户使用产品或产品原型时的行为，并以此找出用户界面中那

些用户难以操纵的部分。

如果以高成本执行测试，那就要雇用一个可用性方面的专家，使用拥有双向镜子的实验室，花几周时间来规划测试过程，找不同的测试者，执行多个测试，对测试结果进行评价和讨论，撰写一份关于所有问题和可行解决办法的测试报告（可能会包含有视频）。

如果以非常低的成本执行测试，你可以把笔记本带到咖啡厅，请别人用你设计的产品完成一些简单的任务。

本章介绍折中方案：以尽可能低的测试成本获得好的测试结果。

可用性专家雅各布·尼尔森认为[1]，即便是差劲的可用性测试也能得出一些有用的结果。

> 较好的可用性测试方法确实能产生更好的结果，至少平均是这样。但记录中拥有最佳表现的测试小组只用了 56%的最佳可用性测试方法。即便是那些只使用了 20%～30%的测试方法的小组（也就那些把用户研究做得很糟糕的人），也能发现产品中 1/4 的严重的可用性问题。
>
> 在设计中找出两个严重的可用性问题非常值得你为此而付出努力，特别是在你只邀请三个用户参与可用性测试的情况下——通常来说，也就是一下午的工作。（……）
>
> 再差的用户测试也比没有强。

因此，即便是不好的测试也比不做任何测试强得多。但在寥寥几条简单的指导原则之下，我们就能以非常小的预算做出非常好的可用性测试。

很多设计师和开发者都不做可用性测试，因为他们认为自己没有时间。他们日程很紧，花一天时间做些其他事情而不是为产品添加一些功能，会让产品开发日程延迟一天。

问题并不是企业没有执行足够好的可用性测试，而是他们根本不进行测试。本章并不介绍如何执行最好的可用性测试，而是介绍如何开展简单、廉价，但能得到有效结果的可用性测试。

此处我们将介绍执行可用性测试的各种方法。某些方法所需的时间比其他方法要多，但每周只花数个小时也能得到优秀的测试结果。

28.2 测试频率

可用性测试就像慢跑：你做得越多，事情就越容易，你也就越擅长做此类事情。专家们通常建议每个月花几天时间来执行测试，但专门划出几天时间来执行可用性测试很难。另外，如果每个月只执行一次测试，你不可能会熟悉它的。你很难留出时间来执行测试，自己对测试也不熟悉，最终会停止进行测试。因此，我的提议简单得多：每周抽出半天时间来进行测试。

① 参见 http://www.useit.com/alertbox/discount-usability.html。

你很快会发现，测试并得到反馈的循环过程会让你上瘾。每周都会发现新问题，然后找出解决方案，你会急切地想知道自己找出的方案是否有效。相比等待一个月才进行下一次测试，尽早看到新设计方案的效果能让你更加兴奋，无论它是否能解决问题、是否会引发新的问题。

28.3　测试者的数量

雅各布·尼尔森有一个非常著名的观点[①]：仅仅五个用户基本上就能找出一个较大团队所能发现的所有可用性问题。但即便是邀请如此小的用户组进行测试，也有很多工作要做。记住，我们的目标是每周只花半天时间进行测试。寻找五位测试者，把他们同时安排在一天进行测试，如果他们来得过早或太晚，一定要有人接待，准备五份相关的文件——半天时间根本无法完成。

相反，每周只邀请一个人，执行一次测试，也能得到有用的测试结果。

在解释对单个用户进行测试的原因之前，我们先看看反对观点。尼尔森在他的文章中指出了这种做法的两个缺陷。

首先，如果只邀请一位用户进行测试，你没有其他任何东西与所得测试结果进行对比。如果邀请五位用户进行测试，你能很容易地找出比较严重的问题，因为有多个用户都在此卡住了。通常，即使只邀请一个用户参加测试，你也可以合理地推测出某个问题的重要性，只是每次都这样做肯定是不行的。但是，这并不是什么大问题，因为你每周都对设计方案进行测试，如果不确定某个问题是否严重，你可以忽略它。严重的问题很快会再次出现的。

另一个问题在于，你费尽心思所准备的测试只产生了一个结果。当执行更为广泛的测试，需要做更多的准备工作时，这一问题尤为明显。非常大的投入只产生了少量的产出。如果测试计划是执行广泛的可用性测试，需要做大量的准备工作，你就要邀请更多的人来执行测试。然而，本章所介绍的测试不需要做太多的准备。

因此，只邀请一个用户进行测试是有一定缺陷的。幸好，这样做也有一些明显的优点。

- 由于每次只需要一位测试者，因此你能较容易地找到人进行测试。
- 你不需要调整多个人的日程安排。为两个人（设计师和测试者）找出一天来共同完成一件事通常是比较容易的。再找另外一个人在相同的时间内进行测试会显著地加大难度。这也说明了为什么我们约见的大部分人的办公室都有会客间：预约见一个人是很难的，约定时间通常也很难奏效。

  即便能把所有人都约在同一天，你在测试过程中也会遇到问题。如果只有一个测试者，你可以把测试时间延长一些，但如果有多个测试者，其他人在等待，你就必须在原计划的时间内完成测试。

① 参见 http://www.useit.com/alertbox/20000319.html。

- 如果只有一位测试者，你可以很容易地自己开展整个测试。如果有更多的用户参与测试，你就需要找额外的人来照顾那些处于等待状态的人，并在你执行测试的时候接待到来的其他用户。

当然，如果时间、预算、协助者都充足的话，邀请三至五位测试者比邀请一位能提供更多的结果。但不能因为时间紧张和预算不足就不做可用性测试。

（注意，这一限制条件不适用于上一章介绍的游击队式的可用性测试。如果你到测试者所在的地方进行测试，而不是让测试者来找你，就不存在邀请多个测试者进行测试的问题。在这种情况下，我建议同时进行多个测试。）

28.4　产品测试对象

除非是针对某个特定群体设计产品，邀请谁来执行可用性测试通常都是可以的。即便你的目标是某个特定的群体，邀请这一群体之外的人来执行测试也是有意义的。为什么这样说呢？原因有两个。

其一，人们的行为存在惊人的一致性。如果你的设计存在有严重的问题，无论让谁进行测试，你都能发现这些严重的问题。

其二，即便你的目标是某个特定的群体，这一群体内的人也不一定具有完全一样的特征。就某一问题，某些人可能比其他人阅历丰富。如果刻意从目标群体中挑选测试者，你会错过那些阅历较浅的人可能会遇到的设计问题。比如对一款在某个医院中使用的产品进行测试。如果你只邀请在那个医院中工作的人进行测试，那么这些测试者对该医院的流程都很熟悉，也熟知相关的医学术语。但那些新来的人可能根本不了解这些东西，他们可能会遇到一些你所挑选的测试小组中的成员都不会遇到的可用性问题。因此，虽然要挑选目标群体中的人进行测试，但不能让测试者的来源仅局限于这一群体。

这同样适用于那些已经参与过某个产品的可用性测试的人群或用户群。如果你要更新某个产品，通常要从那些已经熟悉产品的人群中挑选人来参与测试。这些人已经知道了产品的某些行为，他们对用户界面的操作完全不同于新手用户。你要着重邀请那些对测试没有任何先入之见的人参与测试，但同时也要邀请一些现有用户。

测试中一个非常重要的问题是测试者的文化背景。同一图标和颜色对不同地方的人有着不同的含义。比如，日本的电子游戏通常会用巨大的红色圆圈表示胜利，这一标识看上去有点像停止符号。这一标识对日本用户来说意味着胜利，但在西方用户眼中却表示相反的含义。

如果想让大量有着不同文化背景的人使用你的产品，所挑选的测试者一定要能反映出这种多样性。

虽然本书不涉及为残障人士设计方面的内容[①]，但我仍要说明，让视力较弱的人、老年人，以及那些存在缺陷、会影响到与产品的交互过程的人参与测试是很有意义的。这样做能帮你找出那些以前未被发现的可访问性问题。

主要问题在于：不要浪费太多的时间在某个特定群体中招募测试者。大部分时候，无论背景如何，人们总能发现相同的可用性问题。

28.5　如何寻找测试者

可以通过很多途径找到测试者。你可以尝试以下几种方式。[②]

- **朋友和亲人**。招募测试者最简单的方式就是找朋友和亲戚参与测试。让测试者重复进行测试是 OK 的，但不能让相同的人一次又一次地做相同的事情。我不建议让你的同事参与测试，因为他们对产品的理解比普通用户或多或少要深一些。
- **专业招聘机构**。有很多专业的招聘机构能帮助你找到测试者，但在决定要选择哪个机构之前一定要对比一下服务价格。如果要让专业机构招募测试者，他们找到的可能是那些常常会参与类似测试的人，这些人也是你不想找的。
- **通过你自己的网站招募**。你也可以通过自己的网站招募测试者，但这样做可能会找到一些产品现有的用户。
- **通过当地的报纸招募**。在当地的报纸上刊载一则招募启示通常会产生很好的效果。如果你住在某个大学附近，校园里应该会有公告栏，你可以在那里打出招募测试者的广告。

记住，有些招募方式可能要向测试者付一些费用。

28.6　不同类型的测试

如果你正在开展纸质原型测试，那么执行可用性测试的方式是非常有限的。纸质原型的性质决定了你所能执行测试的方式。用可运行的源代码进行测试拥有更大的自由度，已经有很多关于如何开展此类测试的观点。我将其粗略地分为三种不同的测试：有主持任务测试、无主持任务测试和自由测试。

各类测试的准备工作也各不相同，我首先介绍这三种不同的测试，稍后说说如何开展每种测试。

有主持的任务测试

在有主持的任务测试中，由协助者主持测试，并向测试者介绍不同的产品行为或测试任务。

---

① 如果想要查找这方面的信息，可以参考马克・皮格瑞姆的文章 Dive Into Accessibility，参见 http://diveintoaccessibility.org；还有乔・克拉克的网站 http://joeclark.org。

② 克里斯汀・佩尔费蒂在介绍可用性测试的视频中提出了一些其他方法，参见 http://uxideas.com/shows/usability_tests_nutshell/。

主持人在测试者执行任务时观察其行为。在测试过程中，协助者与测试者待在一起，并与他们进行交流。

无主持的任务测试

在无主持的任务测试中，协助者向用户介绍可用性测试，并解释所有界面元素的工作原理。协助者在纸张上写下一系列任务，交由用户完成，然后离开测试房间。在这种测试中，测试者和协助者之间几乎没有交流，或者交流很少，这样就排除了协助者可能会对测试者产生的影响。

自由测试

在自由测试中，协助者不给测试者分配任何任务，而是鼓励测试者自己探索产品，做自己想做的事情。如果你找到的测试者已经对产品产生了兴趣（也就是说，你是通过自己的网站招募到了这些测试者），这种做法获益最大。

在现实中，你通常需要综合执行以上不同类型的测试。例如，首先让协助者鼓励测试者探索产品，然后转向基于任务的测试。

28.7　测试的准备工作

根据执行测试类型的不同，你需要做不同的准备工作。大多数情况下，你都会用到电脑。卸载掉所有与测试无关、可能会影响测试的东西，比如病毒防护软件。安装产品，保证所采用的安装方式能让你很容易地把产品恢复到初始状态（你肯定不想让测试者因为其他用户输入的数据而感到困惑）。安装屏幕记录软件和麦克风。如果要对测试者进行录像，你还要安装网络摄像头。

尽可能保证产品所在环境的真实性，这一点非常重要。如果能避免的话，一定不要在电脑的模拟器中对 iPad 的应用进行测试，而是应该分配给测试者一台 iPad，然后在 iPad 上对应用进行测试。如果不这样做，你可能会错过一些重要的问题。比如，用鼠标可以点击到的某些很小的按钮，用手指却根本触摸不到。

如果将要测试的软件的运行平台不是电脑，你或许要准备一台能够记录整个测试过程的摄像机。

如果执行的是协助者与用户待在一起的有主持的测试，虽然对整个过程进行录像的做法很有用，但这样做的必要性不是很大。

准备测试任务

对于任务测试而言（无论是有主持还是无主持），你需要为测试者准备测试任务。回想一下本书最前面关于产品行为的内容，你可以参照执行。你要着重关注产品那些最基本、最重要的行为。

程序员用代码覆盖率来度量某个测试活动究竟覆盖了多少代码。在对用户界面进行测试时，你很难度量某个测试过程覆盖了多少用户界面。但你可以把“用户界面覆盖率”作为目标，设计测试任务，使其能够覆盖用户界面的不同区域。

和为纸质原型测试设计任务一样（第 11 章，如果愿意的话，你甚至可以用为纸质原型可用性测试设计的任务进行测试），一定不能让任务有固定的完成方式，因此要设计出用户在实际中可能会遇到的使用情景。描述任务场景和目标，不要涉及任何实际的操作步骤，也不要用那些屏幕上显示的词汇进行描述。

至少准备五项任务。对于有主持的测试，把每项任务分别打印到单独的纸张上。对于其他类型的测试，把任务打印到一张纸上就可以。

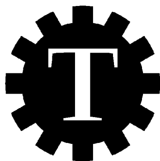
28.8 执行测试

在执行测试时，你通常有两种选择：让测试者执行现场测试，或者开展远程测试。大多数情况下，远程可用性测试要简单得多，成本也低很多，还能避免一些现场测试存在的问题。

我建议先开展一些现场测试，大致了解一下可用性测试的过程，并熟悉与测试者进行交流的过程。如果与测试者坐在一起，完成这些目标会更加容易。下一章，我将介绍如何开展现场可用性测试，之后的一章介绍如何开展远程测试。

提示

- 如果想知道某个用户界面是否有效，你需要让实际用户对其进行测试。
- 可用性测试的成本不一定都很高，也不一定都很耗时。
- 你需要定期开展测试。测试次数越多，效果越好，迭代周期越短，就越容易测试出用户界面是否有效。
- 最好经常开展小型测试，而不是偶尔开展很多人参与的测试。每周让一个人（不同的人）执行一次可用性测试，这一点非常容易做到。
- 让新手用户参与测试，同样也邀请已经知道如何使用产品的人参与测试，因为这两种不同的人很有可能会发现不同的问题。
- 挑选拥有不同文化背景的人参与测试。同时不要忘记开展可访问性测试。
- 可用性测试有很多不同的类型，包括有主持的测试和无主持的测试，既可以是基于任务的测试，也可以是自由测试。通常，你需要综合开展不同类型的测试，首先让用户自己对产品进行探索，然后再开展预先定义好的任务测试。
- 如果正在开展基于任务的测试，你先要准备好测试任务。任务不能有固定的完成方式，描述出任务场景，让用户自己思考解决方案。



第 29 章

现场测试



什么是现场测试?

基本来说，现场测试是指你邀请用户，观察他们使用产品时的情形，并利用所得信息改进产品的过程。

为什么要这样做?

无论执行什么样的可用性测试，你都能找到用户界面的问题所在。

除此之外，即便打算执行远程测试，你也要首先开展一些现场测试，大致了解可用性测试的过程。

是否存在前提条件?

当然存在。你应该先阅读前一章内容（第 28 章）。

执行现场测试

此时，假设你已经找到了一位测试者，也做好了所有测试准备工作，让我们直接进入实际的测试过程。你所要做的第一件事情是向测试者介绍此次测试。与纸质原型测试（第 11 章）一样，对此次可用性测试进行简短的介绍。你可以参考以下这个例子进行叙述。

> 嗨，我是卢卡斯，是 BizTwit 公司的软件设计师。今天，我们将对 Twitter 的客户端 BizTwit 的新版本进行测试，看看它是否能如我们所期望的那样有效。我要郑重声明的是，我们测试的是产品，而不是你本人。这款新产品已经在我们团队之外得到了应用，因此我们想通过观察人们与它进行交互的过程来发现问题。如果你遇到了困难，或者某些东西没有如你所愿般有效，也不要担心——这正是我们所要寻找的问题！在与产品进行交互的过程中，你可以随意说出自己的任何想法。这能帮助我们了解你在使用产品时

大脑中在想什么，我们也能更容易地知道该如何改进这款产品。如果你同意，我们将记录本次测试过程。这样做能帮助我们找到办法修正产品所存在的问题。我们保证不会以任何形式发布此次记录的内容。

你还需要根据将要执行什么类型的测试，介绍如何开展测试。

接下来要做的事情取决于你要执行什么类型的测试。

有主持的任务测试

首先介绍有主持的任务测试将如何开展。向测试者说明他们可以提问，但你并不一定每次都回答。

然后由你来扮演电脑（或设备）。在开始测试之前，电脑应该处于初始状态。你来启动产品（如果所测试的是一个网站，在浏览器的地址栏输入网站的 URL；如果是一个应用程序，启动该程序），然后把控制权交给测试者。

你要知道的第一件事情是，测试者是否能识别出产品是什么。因此，在测试开始时进行以下说明：

请看一下这个网站，你可以随意浏览任何位置的内容。你认为这个网站所提供的是什么样的服务？

如果测试者看不出该网站是用来做什么的，你就发现了第一个可用性问题。

接下来，你想知道测试者是否知道如何使用产品。准备一些测试任务，把每一项任务列在单独的纸上。然后说：

为了测试这个网站，我将分配给你一项任务。首先，我会口头介绍该项任务，然后再给你看纸质的任务介绍。

然后，递给测试者写有任务的纸张，保持沉默，观察测试者的行为。此时可以记下笔记，但尽量不要打扰测试者。不要发出任何噪声，绝对不能发出笑声，坐在测试者后面一点的位置。

如果测试者完成了一项任务，应表示口头感谢，然后进行下一项任务。如果测试者看上去不高兴，或者遇到困难卡住了，你就要停止当前正在进行的任务。如果必须要停止某项任务，一定不能阻止测试者，要明确告知测试者，即便他不能完成任务，测试结果依然非常有用。你可以这样说：

我预感到此处的用户界面有问题，我现在知道该修改什么了。我们已经取得了这项测试任务的效果，让我们进行下一项任务吧。



这是唯一需要打断测试者的时候。如果想要与测试者交流些什么，你可以先记下来，等他完成任务之后再问他。如果测试者有问题，你只要用模棱两可的话回答就可以了：

> 如果你独自在家，遇到这样的问题，你会怎么样处理？

在完成最后一项任务之后，问完所有问题，请测试者对测试过程进行评价，感谢测试者花费时间参与测试，乐观地对测试过程做个总结。

无主持的任务测试

有主持的任务测试存在一些由我们自身引发的问题。作为协助者，我们必须在测试过程中与测试者待在一起，即便我们竭尽所能不去打扰测试者，但依然很可能会对他们产生微妙的影响（更多内容，参见第 31 章）。

协助者还会让测试者感到紧张，谁都不愿意有人坐在后面，持续地观察自己的一举一动，更不想让人将自己的行为记录下来。最终，测试者常常会下意识地讨好协助者，如果我们坐在测试者旁边，他们很可能会一直尝试执行某项操作，期待可以产生作用。

解决这一问题最简单的方法就是离开测试现场。

在无主持的任务测试中，向测试者介绍测试过程，递给他写有几项任务的纸张，然后把测试者留下来，让他自己完成任务，并对整个测试过程录像。

这种方法有个问题，当测试者遇到困难时，你无法很容易地介入其中。在测试刚开始的时候，你就要说明如果发生了类似的情况该怎么办。根据测试设备的不同，需要采用不同的方法解决此问题。如果在安装有双向玻璃镜的实验室开展可用性测试，这个问题很容易解决，你可以在隔壁观察整个测试过程，可以随时与测试者进行沟通。

如果在无法与测试者进行沟通的场所进行测试，你可以这样说：

> 如果你在执行某项任务时遇到困难，不要担心。这正是我们想要发现的情况。你可以自由地进行尝试，但如果感到厌倦，或者觉得自己无法继续完成任务，或者认为产品出现了问题（这完全有可能，因为产品还没有完成），跳到下一项任务就可以了。如果有任何原因使你无法这样做，我就在隔壁，出现了任何问题，都可以随时过来找我。

除了能将主持者产生的影响降到最低，无主持测试还能让测试者感到更加放松，因为没有人坐在他们旁边，时刻关注他们的行为。

自由测试

自由测试中没有测试任务。其目标是观察测试者在不知道下一步应该做什么的情况下的反应。只要向测试者介绍一下测试，然后让他自由摆弄产品，你自己观察将要发生的事情就可以了。我建议采取现场观察或在其他房间观察的方式，这样可以在测试者感到厌倦，或在测试过程中遇到困难时介入其中。如果确实发生了类似的情况，给出一些任务，然后执行基于任务的测试。

是否要鼓励测试者说话？

如果你愿意，告诉测试者在测试的过程中可以随意说话，这是个不错的主意，但不能强制测试者一定要说些什么。有些人本身就不喜欢说话，经常鼓励他们说出自己的想法会让他们感到不自在，也会分散他们的注意力（尤其是如果他们必须用非母语语言交谈，会更加分散注意力，正如有时测试那些拥有不同文化背景的人一样）。不要只挑选那些喜欢说话的人作为测试者，你并不想只让外向的人对产品进行测试。

虽然倾听测试者的心声非常有用，但这不是必须的。大多数情况下，即便你不知道测试者心里在想什么，也能明显看出测试者发现了一些问题。

根据我的经验，如果鼓励测试者说话，他们会提出自己的观点。你的主要工作是关注测试者在做什么，而不是说了什么。如果测试者偏离了测试本身，或者开始提出自己的观点，而不再完成测试任务，那么应礼貌地提醒他们，你希望他们做什么。你可以这样问："你在找什么？"以此把他们拉回到测试过程中。

是否要记录测试过程？

如果执行的是有主持的测试，不一定非要记录测试过程，因为你可以看到测试者在做什么，也在测试过程中记下了笔记。但假设记录不是必需的，你是否仍要记录呢？

即便我记录了测试过程，通常也不会返回去查看录像。但录像很容易制作，成本也很低，并且有时候会发挥大作用。如果录像对你来说很容易，那么就记录测试过程。但如果你无法录像，也无关紧要。

对于苹果电脑，我建议使用Silverback[①]，它是一个专门针对可用性测试开发的屏幕录像软件。用QuickTime播放器也可以制作简单的屏幕录像，选择“文件→新屏幕录像”就可以了。

对于使用Windows系统的PC，TechSmith[②]提供了多种不同的屏幕录像方法。

对于Linux系统，你可以使用类似于xvidcap[③]的软件制作屏幕录像。

如何处理测试结果？

你要列出已发现的问题，并对其进行排序，修正最重要的问题，再次进行测试。如果无法修正所有问题，也不要担心，假如某个严重的问题没有得到修正，它最终会再次出现在后续的测试过程中。

如果有多个人观察测试过程，就对测试过程中的便利贴进行事后检查。让每个人在便利贴上写下自己认为最重要的十个问题。你很快就能找出哪个问题最严重，需要马上修正。

除非真的需要，否则不要撰写测试报告，与只修正问题相比，这样做需要完成更多的工作。通常情况下，没有人会阅读测试报告。因此，你应该找出最紧迫的问题，然后将其修复。

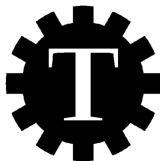
提示

- 每周邀请一个人（不同的人！）执行一次测试。这样很容易安排测试者，也能迅速迭代测试修正问题的过程。
- 开展本章介绍的某种类型的测试，或者综合开展多种测试。让用户自己摆弄产品通常是个不错的办法，你可以观察他们如何与产品进行交互，然后再开展基于任务的测试。
- 以乐观的态度对测试做出总结。
- 用常规方法对测试结果进行评估。找出最紧迫的问题并修正它，然后再次测试。
- 如果不确定某个问题是否广泛存在于用户之中，不要急着解决它。如果该问题比较严重，它会再次出现于后续执行的测试中。

延伸阅读

如果想要了解更多关于现场测试的内容，我建议你读一下史蒂夫·克鲁格的《妙手回春》（*Rocket Surgery Made Easy*）。

① 参见 http://silverbackapp.com。
② 参见 http://techsmith.com。
③ 参见 http://xvidcap.sourceforge.net。



第 30 章

远程测试

30



什么是远程测试？

开展传统的现场可用性测试，你要到测试者所在的地方，或者测试者来找你。安排时间会见人通常很费事，需要做很多工作，以至于经常会延迟见面时间。

幸好，不一定非要与人们见面才能开展可用性测试，你可以开展远程测试。本章将介绍如何开展远程可用性测试。

为什么要这样做？

远程测试通常比现场测试简单，它绕过了可用性测试中那些最棘手的问题，比如在附近寻找愿意参加测试的人，安排他们执行测试等。没有这些问题的干扰，意味着你可以经常开展远程测试，这样能找出并修正更多的可用性问题。

是否存在前提条件？

当然存在。你应该先阅读本章之前的两章内容。

30.1　有主持的远程测试

假设你要开展可用性测试，但在附近却找不到任何测试者；或者你想要让参与测试的人更多样化，所以要邀请住在不同地方的人参与测试；或者你要通过自己的网站招募来自世界各地的人参与测试；或者你只想简化测试过程，避开所有现场测试的要求条件。

那么你可以利用诸如Skype、iChat或Copilot[①]之类的屏幕共享软件来完成测试。你不需要在现实中会见测试者，通过网络会面就可以了。

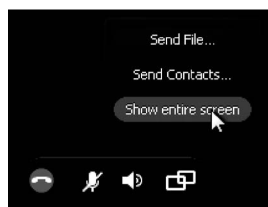
① 参见 https://www.copilot.com。

Skype 能共享屏幕吗?

当然能。但 Skype 的屏幕共享功能有点难找。如果要在 Mac 电脑上启动该功能，首先要开启一个视频会话。然后在视频窗口中点击标有“方块和箭头”图标的按钮，选择“共享屏幕”（Share Screen），再选择要共享的屏幕就可以了。



在 Windows 系统中，启动一个视频会话，然后点击带有“两个矩形”的图标，在弹出窗口中选择“显示整个屏幕”（Show entire screen）就可以了。



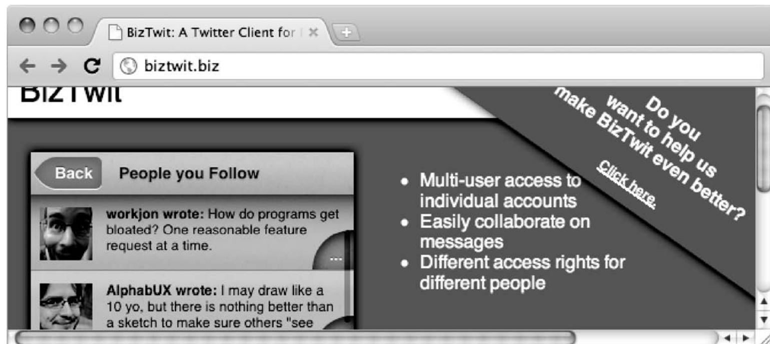
由于测试者必须共享屏幕，所以如果使用 Skype 开展远程测试，你必须帮助测试者完成以上这些稍微有点复杂的步骤。

招募测试者

在执行传统的可用性测试时，你要做的第一件事情就是招募测试者，然后安排测试时间。因此，在测试开始之前你可能要花费几天甚至是几周时间来招募测试者。在远程测试中，你不必安排实际的会面地点，因此，在测试开始之前不必招募测试者。

远程测试提供了一种传统可用性测试无法实现的招募测试者的方式：“现场招募”。在你的网站上发布一条消息：你正在找人测试新软件或网站，然后附加一个指向招募页面的链接就可以了。如下页图所示。

为了增加应征者的数量，你可以为参与者准备一些奖品，比如，一张小面额的亚马逊礼券。



你应该准备一个表格，让访问该链接的人输入他们的联系信息（根据开展测试方式的不同，要求访问者输入 Email 地址、Skype 账号或电话号码等），以及其他需要用来筛选测试者的信息。内特·宝特和托尼·图拉希缪特在他们合著的书《远程研究》（*Remote Research*）中建议用这个表格剔除那些专门为了参与测试才来访问网站的不合格测试者。（只有为测试者提供奖品时才会出现此类问题。）他们建议向访问者提出一些问题，比如“你今天为什么访问这个网站？”来筛选测试者。他们在书中写道：

> 这个问题不仅能帮助你识别出某个用户访问网站的目的是否与此次测试的目标一致，还能帮你剔除掉那些假冒的测试者。（……）大部分真正的访问者都有特定的访问目的，通过询问开放性的问题，你能强烈地感觉到谁才是真正访问网站的人。在开始筛选时，我们通常会提出简单的问题：“你今天为什么访问（某某网站）？”如果访问者的答案模糊不清（“寻找一些信息”、“随便看看”、“看看这里有什么东西”），那么就要进一步进行仔细的审查后才能联系他。如果访问者的答案很明确、很具体，也与此次测试的目标十分相符（我访问该网站的目的是对比 iDongle 和 iDongle Pro 的价格），那么就可以放心地选择他作为测试者。

他们建议用Ethnio[①]——通过网络现场招募测试者的工具来创建招募表格。如过有人填写了表格，你可以立即联系他（如果是合格的测试者），并启动测试。

介绍测试

在开展远程测试时，你和测试者并不在同一地点。因此，你很难和他建立融洽的关系，也很难得知测试者的心情如何。

首先通过电话或视频聊天建立联系，介绍测试流程。所介绍内容与常规的可用性测试相同：说明你测试的是什么，测试者需要做什么事情（前几章已介绍）。确保测试者知道你在测试过程中能看到他的屏幕。如果测试者对此并不知情，或者测试者不太愿意答应你的请求，那么就终止测试，寻找其他人。

① 参见 http://www.ethnio.com。

如果测试者愿意共享屏幕，帮助他完成共享屏幕的操作。（实现屏幕共享所需的详细步骤取决于你使用什么屏幕共享系统。）一定要明确知道测试者必须做什么事情才能访问应用程序或网站。[①]

执行测试

在很多方面，执行远程测试都比现场测试要简单，因为你不太可能会影响到测试者。你只需关注测试者所做的事情，保持沉默就可以了，除非测试者明确地向你提出了问题。对于其他可用性测试，不要引导测试者，如果测试者看上去不太自在，或者有点生气了，那么就停止测试。

由于你和测试者不在同一房间内，也就很难知道用户在何时开始变得沮丧。经常关注一下测试者的语气，以便能在测试失败之前捕捉到测试者的所有沮丧情绪。

在测试的最后阶段，你可以像在传统的可用性测试中一样对测试情况做个总结。询问所有你在测试中遇到的问题，请测试者对整个测试过程进行评价，感谢测试者花费时间参与测试，最后给测试者一些奖励（如果你准备了的话）。

缺陷

与现场测试相比，有主持的远程测试存在一些缺陷。

因为你不在测试者身边，所以会错过一些现实中的线索。测试者是否开始变得沮丧或是恼怒？进行远程测试的时候对此很难判断，因此你要特别注意，如果测试者显得疲惫或是生气，那么就要准备停止测试。注意测试者的语气和措辞。

另外，你也无法看到测试者在关注什么。如果你不知道测试者在做什么，或者不确定测试者是否仍在执行测试，你可以问他“你在看什么呢？”，或者告诉测试者，如果他能用鼠标点击一下自己正在关注的东西，将会对你有所帮助。

如果你与测试者不在同一房间内，那么让测试者签署同意录像的表格还需要执行另外一些步骤。你可以在测试开始前通过邮件把表格发给测试者。然后在测试开始前打电话给他，复述测试者将要签署的主要内容，并征得他的口头同意。这些步骤是否足以帮你征得测试者的同意，取决于你们当地的法律。

在执行远程测试时，你通常无法控制测试者所在的环境。带来的缺点是运行产品可能会需要一些时间，优点就是你可以观察测试者在自己的环境中与产品进行交互，此时测试者的行为要比在受控的环境中更为自然。另外，这样做也能让你看到产品在一般的电脑上运行的情况，而不是在你通常所用的高端电脑上的运行状况。测试者有可能会在测试过程中走神，比如，停止测试去接电话或查看新收到的邮件。如果可能，请测试者关掉手机，关掉所有聊天程序，并关闭邮件的周期性提醒功能。

30

① 如果所要测试的产品不会出现在测试者的屏幕上，比如 iPhone 的一款应用，请测试者把网络摄像头对准产品，这样就能看到他所做的事情。

因为不能按照自己的意愿搭建测试环境，所以测试之前你要帮助测试者完成一些事情。查看测试者所用的电脑系统是什么，他都安装了些什么程序，哪个版本。（如果是通过网站招募测试者，你可以通过小规模的问卷调查完成这一任务。）一定要知道如何帮助测试者准备好所有的东西，并让产品开始运行。还要确保测试者的网络连接速度足够快，能够支持开展远程测试。

除了让测试者共享他们的屏幕之外，你还可以用自己的系统进行测试，然后把你的屏幕共享给测试者。有时候确实需要这样做，因为你无法保证产品总能在测试者的电脑上运行。但如果可以，尽量不要这样做。把你的屏幕共享给测试者会降低他那一端所看到图像的质量，系统也会在输入和反应之间有一定的延迟，很多人都非常讨厌这一现象，它会毁掉测试结果的。

优势

我已经提到过，远程测试的优点之一是，测试者可以在自己的环境中对产品进行测试，但这并不是远程测试的唯一优势。

另一个非常大的好处是，如果你的网站拥有一定量的点击率，远程测试很容易招募到测试者。

另外，它可以完全消除你对测试者产生影响的可能性。远程测试者看不到你的面部表情和身体姿势，他们不会受到你的行为的影响。

当然，为远程测试安排测试者也很容易，你不需要在某个实际的地点会见测试者，也完全不必为所有的测试者安排时间。

测试示例

为了向读者说明典型的远程测试是如何开展的，以下是一个假设的、稍微有点简略的测试。

对于这个测试，假设测试者在你提供的在线表格中填写了自己的 Skype 账号名称，此时你正通过 Skype 和他取得联系。一定要戴耳机，这样才能空出手来记笔记。

> **协助者：**嗨，我是卢卡斯·马希斯，来自 BizTwit.biz。你好！大约 20 分钟以前，你填写了我在网站上提供的表格，说愿意帮助我们改进 BizTwit。
> **测试者：**是，是的。真不敢相信你联系了我！
> **协助者：**那么，你能否空出 15 分钟时间来执行测试呢？
> **测试者：**可以，当然可以。

如果测试者无法腾出时间，那么就停止测试。

下一步，我们要确认当前这个人是否是我们想要找的测试者。

> **协助者：**你以前用过我们的 Twitter 应用 BizTwit 吗？
> **测试者：**没有。
> **协助者：**你是因为对这款应用感兴趣才访问我们的网站的吗？
> **测试者：**是的。我正在寻找一款商用的 Twitter 应用，然后在 Google 上找到了你们。

协助者：很好，你正是我们要找的人，下面我来介绍一下此次测试。我们正在对这款应用的新版本进行测试，想知道用户是否知道该如何用它，进而找出需要改进的部分。我们要测试的是新设计方案的效果，而不是你本身。为了开展测试，你必须向我共享你的屏幕，你觉得可以吗？这样，在整个测试过程中，我就能看到你的屏幕上发生的事情。

测试者：可以，没问题。

协助者：好的。从现在开始，我将记录这次测试过程。我们将通过录像来寻找设计中存在的问题。我们不会将它公布于众的，仅仅用于改进产品而已。

测试者：好的。

如果测试者在回答最后两个问题时有所迟疑，或者不太确定，那么就停止测试。如果所有的事情都 OK，就可以开始记录测试过程。

协助者：很好。在开始执行测试之前，我已通过邮件发给你了一个同意进行录像的表格。你收到这封邮件了吗？

测试者：是的，几分钟之前收到了。

协助者：看过了吗？

测试者：是的，浏览了一下。

协助者：OK，我们再迅速浏览一下要点，这样你就知道自己签署了什么东西。

明确地介绍测试者所要签署的内容，并征得他的口头同意（根据你所在地的法律条款，履行所需的其他手续）。

协助者：好的，所有要准备的东西都就绪了，让我们开始测试吧。首先，你要向我共享屏幕。从现在起，我能看到你的屏幕上的所有内容，直到测试结束。如果屏幕上有任何私人文件，比如邮件，现在请把它关掉。

向测试者解释如何共享屏幕。

协助者：好极了，现在我能看到你正在做什么。首先，我们要在你的电脑系统上测试 BizTwit 的最近版本。请访问 biztwit.biz……

解释如何启动产品。如果应用程序需要登录，而人们可能不太愿意输入自己的登录信息，那么你就要预先建立一个测试账号。

协助者：现在 BizTwit 已经在运行了，你认为它如何？它看上去像是你愿意使用的软件吗？只通过观察该应用，你能否比较准确地猜出它的作用、功能的使用方法呢？

简短地介绍一下产品，询问测试者对产品的第一印象，然后转入基于任务的测试。

协助者：OK，我们开始执行第一项任务。我将向你介绍该项任务，在执行任务的过程中，你可以说出自己的想法，告诉我你正在想什么。如果有问题，你可以提问，但

30

我可能不会做出详细的回答。这只是测试的一部分内容。在任何时候，如果某些无效的东西让你感到厌烦或是不安，你都可以停止测试，好吗？

测试者：好的。

给出第一项任务。关于如何设计测试任务，参见前几章和第 11 章。

测试任务一旦完成，首先解释如何停止屏幕共享，然后可以就任何问题进行提问。

最后，对测试进行总结。此处的目标是以乐观的态度完成测试。被人看到犯错误会让测试者感到紧张，告诉测试者他们所做的事情对你非常有用，这样能消除对测试者的负面影响，也能让测试者感到自己的付出是有价值的。

协助者：现在，我已看不到你的屏幕了。很好，此次测试帮助我们找出了很多问题。通过今天发现的东西，我们能极大地改进 BizTwit 的用户界面。感谢你对我们的帮助！你是否有其他问题或是意见呢？

测试者：在第一次共享屏幕时，我有点不知道该怎么做，但最后还是实现了共享。

协助者：很好，我下次会以更好的方式解释共享屏幕的方法。我们承诺会奖励给你一张亚马逊的礼券，如果你觉得可以，我稍后会通过邮件把它发送给你。

测试者：当然可以。

协助者：太棒了。如果你有其他的问题或是关于此次测试的看法，可以给我发邮件，我会把邮件地址写在 Skype 的聊天窗口中的。再次感谢你的帮助，祝你愉快！

如果承诺了有奖品，不要忘了发送给测试者。

你不需要撰写任何报告，也不要创建影片来展示测试结果。在测试完成之后，列出你认为最重要的问题，并在下次测试之前修正这些问题。如果你不确定某个问题是否重要，现在可以略过它，看看其他测试者是否会在后续的测试中再次遇到该问题。如果某个问题比较严重，它终究会再次出现的。

是否要执行远程测试？

把所有相关的内容考虑在内，就从测试中获取有用的数据而言，有主持的远程测试和常规可用性测试相比如何？

总体来说，远程测试能否比常规的现场可用性测试产生更好的结果呢？这是可用性领域内激烈争论的话题之一。远程测试或许不如现场可用性测试有效，但它绝对比不做任何测试要强得多。远程测试很容易实施，因此你可以更有规律地开展更多的此类测试。

30.2　无主持的远程测试

一种更为简单的方式是，在完全不与测试者进行交流的情况下开展远程测试。你可以让测试者在使用产品时，或者执行某些预先设定的任务时自己录制屏幕录像。

当然，这种方式也有其优势和缺陷。在无主持、基于任务的远程测试中，如果用户遇到了困难，或者感到沮丧，你将无法介入其中。但另一方面，你可以看到用户自己使用产品时的行为，这更接近于用户在现实生活中使用产品时的行为。

无主持的测试的另一个优势在于，你完全不需要安排任何事情。向测试者介绍测试流程，然后让他在自己有时间的时候执行测试就行了。实际上，你甚至可以利用在线远程测试服务[①]，它能帮助你在测试之前招募测试者。

提示

- 如果还没有执行任何可用性测试，你首先应该开展几次传统的现场可用性测试。你可以先邀请朋友担当测试者，直到熟悉了测试流程。
- 只要熟悉了可用性测试的流程，你就可以开始执行远程测试了。
- 每周至少执行一次远程测试。
- 除了其他渠道（比如邮件列表、讨论小组甚至是报纸），你也可以利用网站招募测试者。利用网站招募测试者极其方便，也不需要任何费用，但如果只通过这种方式招募，你的测试结果可能会向那些有产品使用经历的人倾斜，因为他们很可能访问过你的网站。
- 剔除那些纯粹奔着奖品来的测试者。
- 在联系潜在的测试者时，一定要让他知道你在测试中能看到他的屏幕，并征求他同意对测试进行录像。
- 在执行远程测试时，你通常无法看到测试者的面部表情和他们所关注的东西。注意测试者变得沮丧或不安的情况，如果发生了此类情况，请停止测试。
- 在测试的最后，感谢测试者为此付出的时间，向他说明你再也看不到他的屏幕了，并赠送奖品（如果承诺了的话）。用乐观的态度结束测试过程。

延伸阅读

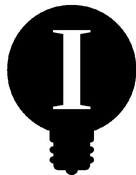
内特・宝特和托尼・图拉希缪特的《远程研究》中包含有大量关于远程测试的有用信息。如果打算执行远程测试，你应该看一下这本书。[②]

内特・宝特还在A List Apart网站上刊载了一篇关于远程测试的文章。[③]

① 提供此类服务的网站有 http://usertesting.com 和 http://fivesecondtest.com。但他们只能为网站的测试提供服务，对于桌面应用程序或移动应用程序的测试则无能为力。

② 参见 http://www.rosenfeldmedia.com/books/remote-research/。

③ 参见 http://www.alistapart.com/articles/quick-and-dirty-remote-user-testing/。



第 31 章

如何避免测试中的常见错误

可用性测试对错误极其敏感。无论你开展的测试糟糕到何种程度，都能够从中得到有效的信息。或许你在测试中并没有找到最重要的问题，或许你只发现了一小部分问题，都不影响你从测试中找到能改进产品的有效信息。

当然，如果你能花些时间开展适当的测试，将能得到更为有效的测试结果。

我在前面几章中曾谈到过这一点，但我认为在此处简要重复一下是有益的。

31.1　不要使用用户界面中的词

在设计测试任务和与测试者交谈时，一定不能使用应用程序中的术语，这一点非常重要。你想要知道的是用户是否能使用产品，而不是他们是否能在用户界面中找到某个特定的词。

假设你正在测试一款文字处理软件，想看看用户是否会使用该软件的拼写检查功能。图 31-1 所示为通过软件菜单访问这一功能的方法。

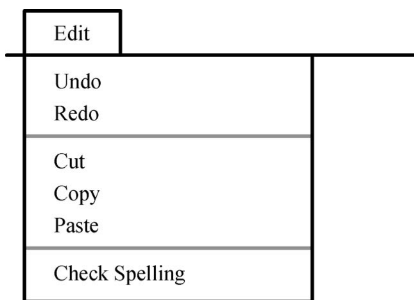


图 31-1　文字处理软件菜单中的常见词汇

如果把任务描述为“请检查你的文档中的词汇拼写是否正确”，那么测试者就会在程序的菜单中搜索，寻找该句话中的某个词。最好的做法是不要描述任务本身，而是描述任务目标：“你要确保文档中没有打错字。”

这样的任务描述更接近于用户在实际中要实现的目标，测试者的行为才能更接近于实际用户的行为。

31.2 不要影响测试者

威廉·冯·奥斯滕是 19 世纪早期的一位德国数学教师。他有一项爱好：驯马。你或许能够猜到他的故事：他试图教会马算算术。让所有人吃惊的是，他的马（姑且称之为“聪明的汉斯”）很快掌握了有一定难度的数学计算：加、减、乘、除和日期计算。这匹马能理解用纯德语提出的数学问题，也能理解写在纸上的数学问题。

当然，马无法把答案写出来，而是通过马蹄踏地的次数给出答案。

心理学家奥斯卡·普丰斯特对此持怀疑态度，并决定对聪明的汉斯展开调查。很快，他就明白了事情的真相。聪明的汉斯其实根本不会做任何数学计算，它根本不会阅读，也无法理解德语，它只是对训练者冯·奥斯滕不自觉的肢体语言做出反应。真正解决数学问题的是冯·奥斯滕，马只是看着他，并用马蹄踏地，直到冯·奥斯滕向马做出暗示，已经给出了正确答案。

有趣的是，冯·奥斯滕却完全不知道自己向马做出了暗示。

这极大地激起了普丰斯特的兴趣，他继续就此问题开展了实验。最终，他发现人与人之间也存在类似的交互过程，即便我们清楚地知道它的存在，也无法抑制这种无意识的暗示。

但这一现象意味着什么呢?

这意味着在测试过程中，协助者会无意识地引导测试者，从而导致测试失效。这也说明了为什么正式的可用性测试通常都在双向玻璃镜后面开展，因为这样做能消除外部因素影响到测试者的任何可能性。

你可以通过一些方法来改善这种情况。首先，即便你无可避免地会给出一些不自主的提示，但可以降低这些提示出现的次数。因此，如果没有绝对的必要，就不要与测试者交流。

其次，你需要从测试者的视野中消失，坐在测试者后面一点，这样他可能不会注意到你的存在。

再次，如果可能的话，在开展测试的时候，不要与测试者坐在一起。远程测试能很好地做到这一点（参见第 30 章）。

最后，不要有太多的顾虑，关注测试者的反应就行了。你并不是在做统计学上的双盲研究，而仅仅是寻找用户界面中存在的问题。如果测试者犹豫不决，或者想寻求你的指导，你就知道此处的用户界面已经出现问题了，即便你给出的答案能让用户继续完成任务。

31.3 避免营造紧张的氛围

所有学医的学生都在医学伦理课上学过这一主要戒律：“首先，不要做对病人有害的事情。”

这一准则同样适用于可用性领域。在开展可用性测试时，我们把测试者置于希望他们犯错的境地。但犯错误会营造出紧张的氛围：没有人喜欢犯错误。你可以做一些工作，让测试者不会因为自己所处的境地而紧张。

第一，明确清晰地指出，你测试的并不是他们，而是用户界面。

第二，如果测试者遇到了困难，很明显地走到了错误的路线上，或许会因此变得焦虑。此时你要介入其中，明确告知测试者，他不会因此受到责怪。你可以这样说："这正是我们开展此次测试的原因。我们的设计还存在有一些严重的问题，我想你得重新考虑一下该怎么做。"然后测试下一项任务。

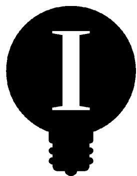
第三，不要重复告诉测试者，他可以大声说出自己的想法。在测试开始时说明这一点是可以的，比如你可以这样说："在测试过程中，你可以大声说出自己的想法，这能帮助我更好地找出用户界面的问题所在。"如果看不出测试者正在做什么，你也可以询问："你正在想什么呢？"但要记住一点，有些人不太喜欢自言自语。

第四，如果测试者看上去开始因为测试过程变得紧张，就停止测试。如果强迫一个焦虑不安、无法集中精神进行测试的人继续进行测试，除了可能会为测试者带来糟糕的经历之外，你得不到任何有价值的东西。他的行为不能代表真正的用户，如果产品惹恼了实际使用产品的人，他们可以休息一会，或是做其他事情。

最后，要以乐观的态度结束测试，感谢参与者为此付出的时间。告诉测试者，他们的参与对你有很大的价值，能帮助你改进产品。

提示

- ❑ 在设计测试任务，或者在可用性测试中与用户交谈时，一定要避免使用用户界面中的词汇。
- ❑ 不要在测试中影响测试者。
- ❑ 如果测试者寻求你的指导，一定要小心，这表明产品出现了可用性问题。
- ❑ 不要让测试者感到紧张。
- ❑ 不要经常要求测试者大声说出自己的想法。
- ❑ 如果测试者看上去焦虑不安或是感到厌烦，就停止测试。



第 32 章

用户错误即是设计错误

当用户无法使用产品时，我们总是把责任归咎于用户的能力不足，通常会说这是“用户错误”。这一错误被简称为“PEBKAC”，其全称为“键盘和椅子之间的错误”（Problem Exists Between Keyboard And Chair）。

在开展第一次可用性测试时，听到负面反应是很自然的。当然，你的产品还没有那么差。可能你只是找了一些非常不称职的人来执行测试，不是吗？

用户自己也会经常感到内疚，他们还会为产品的问题而自责。当用户不知道如何使用产品，你给他们做出演示时，他们常见的反应是：“哇哦，我不知道为什么自己没有想到这样做！现在看上去太明显了！”他们会因为自己没有看到某些东西而自责，而不是把责任归咎于你把某些东西设计得不明显。

因产品的错误而责备用户，根本无益于修正这些错误。如果有上万的用户使用你的产品，其中好几百人，甚至是好几千人都会遇到所有在可用性测试中发现的问题。不要责备用户，你应该修复问题。唐纳德·诺曼在《设计心理学》[①]一书中说道：

> 不要认为是用户犯了错误，而要把用户的行为当做向目标的靠近。

用户没有出错，出错的是你的产品，因为它不能正确地解读用户的操作行为。

32.1　不要在错误信息中责备用户

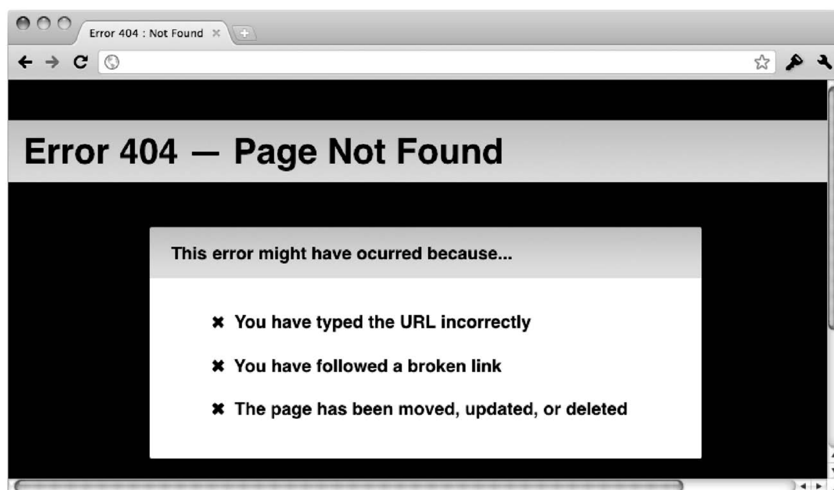
我们经常会遇到如下所示的错误信息：

用户首先看到的是“你输入了错误的地址”。这一错误信息把责任归咎于用户尝试访问无法链接的 URL。但用户可能根本没有做出任何错误的事情。当用户遇到“无法找到页面”时，原因很可能是他点击了一个已失效的链接，或者是网站页面上提供的一个已失效的链接。

不要责备用户，我们应该因为问题向用户致歉。这才是良好的可用性，因为它能让用户冷静下来，处理问题。在《你会对你的电脑说谎吗》一书中，作者克里夫特·纳斯谈到了一个实验。

① 如果还没读过《设计心理学》，那么你应该放下手头这本书，去读一下唐纳德的这本著作。

在该实验中，电脑在用户玩游戏时提供“情感支持”。他总结道：“主动识别并处理用户的情感状态，能缓解挫败带来的强烈负面情绪和刺激。也就是说，当你告知用户已经听到他们的声音、理解他们的感受并表示同情时，他们会稍微好受一些。”



前面所示的错误页面的另一个问题在于，它在描述问题时使用了很多术语。大部分人不知道也并不关心 404 和 URL 分别表示什么意思。

最后，错误页面本身对用户毫无帮助。它描述了产生问题的原因，却没有给出任何解决办法。基于可获得的信息，网站能够测定用户想要访问什么信息。例如，指向博文的链接通常应该包含有撰写博文的年、月、日信息。即便 URL 的尾端信息被截断，或者 URL 的输入不正确，网站也可以利用已输入或正确的 URL 信息判断用户可能想访问的页面。你觉得以下这个方案如何？



这个页面并没有把责任归咎于用户，而是说明该网站可能存在错误，并为出现的问题向用户致歉。它以用户能理解的话语解释了错误产生的原因，还给出了解决问题的办法。如果这些都没用，它还向用户提供了可以直接联系开发者的方式。

用户不愿受辱

在 folklore.org，安迪·赫茨菲尔德描述了在对苹果的 Lias 电脑开展可用性测试时遇到的一个非常有趣的经历。*他解释道，对话框中的两个默认按钮分别是“Cancel”和“Do It”。在可用性测试中，有数个用户在本应该点击“Do It”时点击了“Cancel”。当问到一个这样做的用户时，该用户说：“我不是傻瓜（DoIt），为什么软件把我叫做傻瓜？”

用户没有注意到“Do”和“It”之间的空格，把“Do It”看成了“DoIt”。最后，Lias 的开发团队把“Do It”改成了“OK”。

* 关于完整的故事，还有很多其他与 Mac 电脑开发相关的有趣故事，参见 http://www.folklore.org/StoryView.py?projec= Macintosh&story=Do_It.txt&sortOrder=Sort%20by%20Date&detail=medium&search=Do%20it。

32.2　没有错误就没有责备

不因为用户犯错而责怪他们，这是个很好的做法。更好的做法是在错误信息中给出帮助用户解决问题的方法。但最好的做法还是根本就不让错误出现。如果某些东西出错了，通常并不是用户的错误，而是你的问题。如果产品以另外的方式工作，可能问题根本不会出现。

只需改变用户界面，你就可以防止很多常见错误出现。让我们看一下引发常见错误的两个主要根源。

模式错误

我们所谓的“用户错误”通常可以归咎于产品没有明确显示出当前所处的状态，或没有明确告诉用户该执行什么样的操作。在第 20 章中，我曾讨论过这个问题。我们通常会把某些常见错误认为是“用户错误”，模式正是引发此类问题的一个常见根源，让我们迅速回顾一下这一内容。

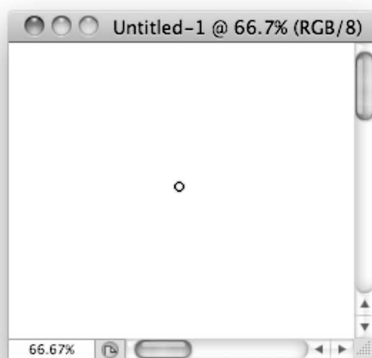
一个典型的模式错误的例子是，手机会在播放影片时出错，大部分手机都没有明显的“静音”模式。如果仅通过肉眼观察，很难识别一个手机当前所处的状态。人们很容易会忘记手机当前是否处于“静音”模式。

像 iPhone 和 webOS 一样，在设备上设计一个明显的硬件按钮来切换这两种模式，就能防止这类问题的发生。

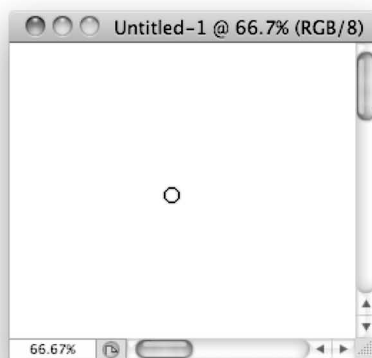


电影看到30分钟后，这两个手机中的一个会产生大量噪音。你能说出是哪一个吗？

如果模式化程序不能明确地告知用户当前所在的模式，或者用户无意之中改变了当前模式，也会出现类似的问题。例如，图片编辑程序通常都包括一个模式化工具，它通过鼠标指针的变化来告知用户当前所选择的工具。用户很容易会忽略这一暗示，比如，当用户认为自己当前选择的是克隆工具（而不是着色工具）时，点击一幅图像却画出了一条线。



点击画点

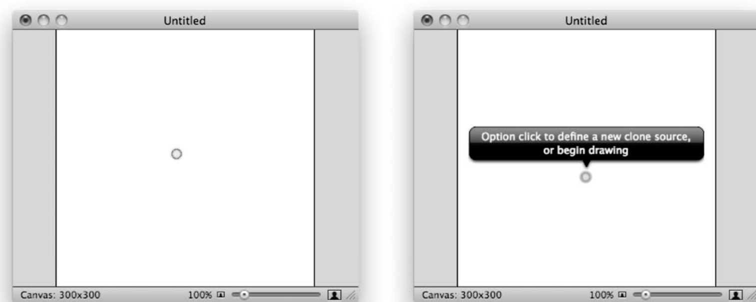


点击复制部分图像

为不同的工具使用不同的指针图像可以解决这一问题。以下是同样的两个工具在Acorn[①]中的显示结果（见下页图）。

Acorn 告知用户他们当前正在使用的工具是什么。这样会大大减小用户对当前所用工具产生的迷惑。

① 关于 Acorn 的更多内容，参见 http://www.flyingmeat.com/acorn。



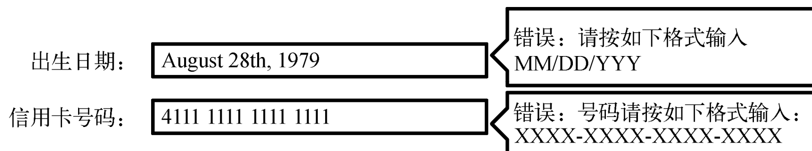
输入错误

当产品没有明确告知用户该做什么时，会出现另一种常见的“用户错误”。比如，过分热心和没有明显提示的数据有效格式：

出生日期：

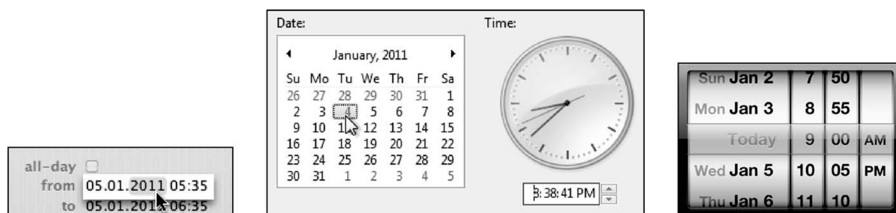
信用卡号码：

产品没有明显地告知用户应该输入什么格式的日期和信用卡号码。因此，用户会尝试输入所有不同的数据格式。这样的设计通常不能保证产品正确地工作。



应用程序和网站应该清晰地表明期待用户做什么（例如，在输入字段旁显示输入内容示例），或者允许用户以自由的格式输入内容。

解决这类问题的另一种办法是，不要让用户决定该在界面中输入什么样格式的数据。比如，不要让用户在文本字段内输入日期数据，而是提供一个日历供用户选择。



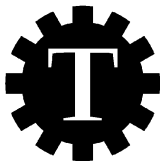
iCal只允许你一次改变一个数字，这样能保证用户输入的数据格式总是正确的。

Windows中的日历程序允许你通过点击数字改变日期，只有在改变时刻时才用输入文本。

iPhone使用了完全避免用户输入的用户界面来改变日期设定。

提示

- 对错误负责。“用户错误”实际上是用户界面设计的失败，是设计师的错误。认为用户应该对错误负责并不能消除问题。
- 不要在用户界面中责怪用户，否则用户会很难受，而且问题可能是由你的设计引起的。
- 提供有效的错误信息。如果某些东西出错了，解释出错的原因，但更为重要的是告诉用户该如何解决问题。
- 避免出现问题。最好完全避免出现错误。如果用户遇到了错误，考虑一下该如何改进产品以避免再次出现错误。
- 避免使用模式，或清晰地说明产品当前所处的模式，这样能防止模式错误。
- 清晰地说明希望用户做什么，或者永久性地阻止错误输入，这样能避免输入错误。



第 33 章

A/B 测试

33



什么是 A/B 测试？

目前为止，我已主要介绍了一些观察用户行为的可用性测试。它们能很好地帮你找出设计存在的问题，但如果要对比两个（或更多）设计方案的优劣，这些测试却难以胜任。

假如你正在设计注册页面上的文字内容，但是不确定什么样的内容更具说服力，能让访问者尽快注册；或者你正在设计主页上的登录表格，目前有两个方案，你不确定哪个方案对用户最有效——解决这些问题最常用的方法是执行 A/B 测试。通过这种测试，你能在两个（或更多）设计方案中找出最有效的那个方案。

虽然我们称其为 A/B 测试，但这种测试不仅能用来对比两个不同的设计方案，而且还可以对比一个设计方案的数个不同版本。因此，这种测试有时被称为 A/B/n 测试。

为什么要使用这项技术？

A/B 测试可以让你用精确的数据改进设计，没有任何推断和猜测的成分。如果你要改进设计方案的表现，A/B 测试正是能帮助你实现这一目标的工具。

是否存在前提条件？

当然存在。你应该准备好能运行的产品版本，还需要找足够多的人来使用它。

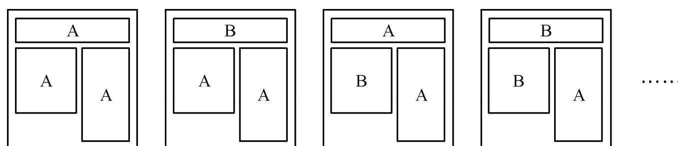
多元测试

多元测试（有时被称为“多变量测试”）是 A/B 测试的一种类型。在执行常规 A/B 测试时，你测试的是已完全实现的不同设计方案。



A/B测试对比的是两个已完全实现的设计方案

在执行多元测试时，你是对设计的各个部分的不同版本同时进行测试。（你的设计方案中有数个变量，因此这种测试被称为多元测试。）



多元测试可以对比设计的各个部分

多元测试其实是并行地对设计方案的各个部分执行 A/B 测试。除非你有足够的理由执行多元测试，否则你最好不要动这种心思，因为这种测试会导致弗兰肯斯坦式的设计，不相匹配的设计元素搅和在一起：某些单独的设计元素虽然能在多元测试中有不俗的表现，但将这些元素放在一起并不一定能形成杰出的设计。一般来说，你最好对比两个或更多已完成、前后连贯的设计方案，看看哪个方案更好一些。

33.1　何时执行 A/B 测试

A/B 测试能很好地比较两个或多个设计，找出最有效的方案。通常来说，适用于以下两种情况。

- 你已重新设计或重新撰写了某些东西，想知道新方案或新内容是否比旧方案更有效。
- 某一问题有数个可行的解决办法，你想找出哪个办法最有效。

你可以用 A/B 测试比较任何东西，从有着显著差异的布局设计方案，到色彩或文字的细微变化都可以。

如果你的网站像 Google 和亚马逊一样，每天都有无数人访问，即便是可用性方面非常小的变化，都可能导致成千上万人遇到问题或避免问题。可能你的产品没有那么多的用户，因此，可

用性方面较小的差异产生的影响不大，那么你就不必通过 A/B 测试来检测产品非常小的变更。但如果变更较大，或变更的区域是产品非常重要的部分（比如，网站的注册页面），那么 A/B 测试再有用不过了。

33.2　什么是“成功地使用产品”

A/B 测试的主要理念是实施两个不同的设计方案，看看哪一个更有效。但这种测试有个与生俱来的问题——“更有效”是什么意思呢？什么样的状况才能表明用户已成功地使用了产品呢？有时候，这些问题的答案很明确。例如，在设计检出系统时，如果用户能完成检测过程，那么系统就是“有效的”。但是通常，对于什么样的状况表示“成功”，并没有明确的答案。

定义“成功”的一种方法是回到设计过程的最初阶段，考虑用户的目标是什么。如果你知道用户使用产品的原因，那么就可以对成功做出定义：如果更多的用户能实现他们的目标，那么产品就更有效。

有时，定义成功要做的事情只是计算有百分之多少的人点击了网站某个链接，或者有百分之多少的网站用户注册了账户。有时候，回答这一问题需要考虑更多的东西。执行任务的人有多少真正完成了任务？对成功的定义取决于你的产品是什么。

33.3　测试的准备工作

此时，你已有了两个（或更多）要测试的设计方案，而且对“成功”也有了一个定义。那么现在，你就要实施这两个设计方案，并想方法分配给用户进行测试。

如果你的产品是桌面应用程序，要么创建两个（或多个）不同的 build 版本，将适当的 build 版本交由各个用户进行测试，要么在程序中同时实现两个设计方案，然后决定在用户启动程序时把哪个方案呈现给用户。

如果产品是网站，你需要实施一种访问网站不同版本的方法，要么给出两个不同的URL，要么决定在用户访问单个URL时呈现哪个设计方案。现在已有很多在线工具支持对网站执行A/B测试。你可以用Google Analytics[①]执行A/B测试，Google还提供了可以用来测试网站变更的Website Optimizer[②]。Vertster[③]也是一个能帮助你对网站执行多元测试的工具。

① 关于如何用 Google Analytics 对网站执行 A/B 测试，你可以阅读乔治·帕默尔的文章，参见 http://www.rowtheboat.com/archives/39。

② 参见 http://www.google.com/websiteoptimizer。

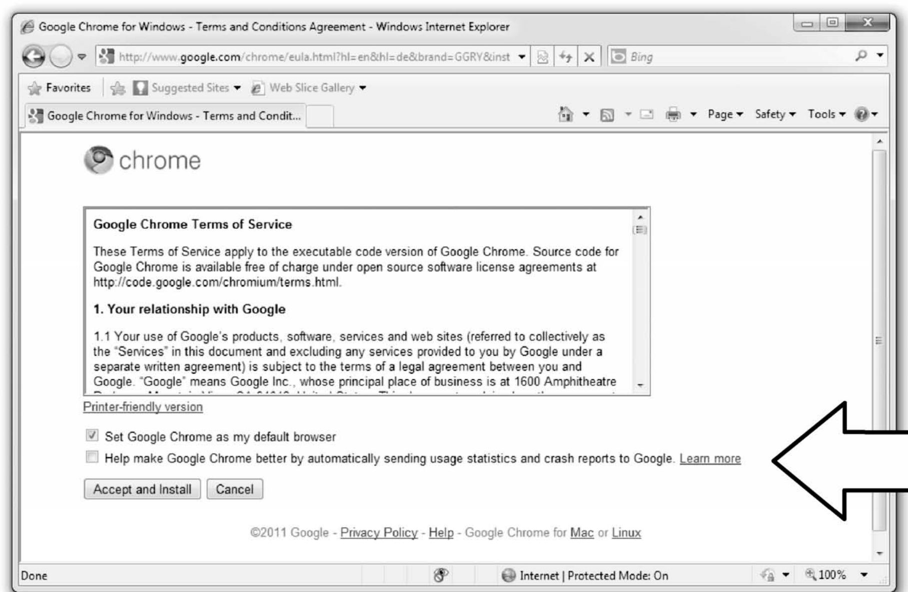
③ 参见 http://www.vertster.com。

33.4　执行测试

在执行 A/B 测试时，你要确保用户不会在两个不同的设计方案间来回转换。（用户正关注某个设计方案时，点击一个链接或重新启动程序之后，突然间又看到别的设计方案，就会感到迷惑。）因此，你应该把用户分成不同的组，确保他们能持续关注某个方案。

在执行测试的过程中，你要有收集测试结果的方法。如果测试的是网站，收集结果很容易（因为你能"看到"每个用户正在干什么），但如果测试的是桌面应用程序，那就要困难一些了。

你还要告知用户，你正在收集产品使用数据，并具体说明你要收集些什么，还要说明这样做的原因。以下是 Google 在 Chrome 的下载页面上的做法。



33.5　解释测试结果

大部分 A/B 测试工具都会解释测试结果，很多时候，你可以提前看出哪个设计方案最有效。但一定要小心：人类是模式的追求者，当没有模式时，人们总能很容易地找出一个来。此时要记住的一点是统计显著性，你需要知道正在关注的测试结果出现的随机概率有多大。

User Effect公司[①]有一个简单的计算程序[②]，该程序能显示出对两个设计方案执行A/B测试所

① http://www.usereffect.com。
② http://www.usereffect.com/split-test-calculator。

得的结果是否具有统计意义。如果两个设计方案的测试结果都不具备统计意义，该计算程序还能告诉你，再需要多少访问者才能得出具有统计意义的结果。

如果 A/B 测试中有两个以上的设计方案，你可以对比各个设计方案，看看差异性是否明显。

33.6　需要记住的要点

如果你用 A/B 测试来对比设计方案较小的、递增式的变化，一定要记住，A/B 测试只能为你带来局部的最大可用性。更多的递增式变化能为你带来好的设计方案，但却不一定能带来最好的。

用户分组

在对桌面程序进行测试时，把用户分成两组相当简单。在用户下载程序时，随机分配给每个用户不同的 build 版本。一般来说，每个用户只会下载一次程序，所以你不必担心同一用户会在不同的设计方案中左顾右盼。同样，如果两个设计方案在同一个 build 版本中，在用户启动程序时，随机地呈现某个设计方案，并把该方案存储为用户的设定。

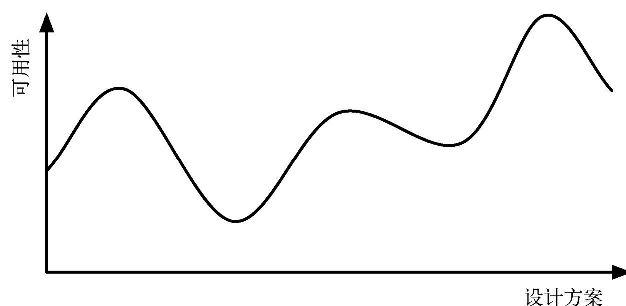
在测试网站时，对用户分组要稍微复杂一些。一种方法是根据访问者 IP 地址的散列模数，为每个访问者选择一个设计方案*，然后设定一个 cookie 显示用户电脑上显示的是哪个方案。对于重复访问者，首先检查他们的电脑上是否有 cookie，如果没有，返回到 IP 地址，决定要显示哪个设计方案。这种系统不能完全防止错误（用户可能会通过更改 IP 地址禁用 cookie），但几乎能覆盖所有情况。

* 由于 IP 地址并不随机，仅仅获取 IP 地址的模数，可能得不到不同分组访问者的人数，把散列函数应用于 IP 地址就能解决这一问题。

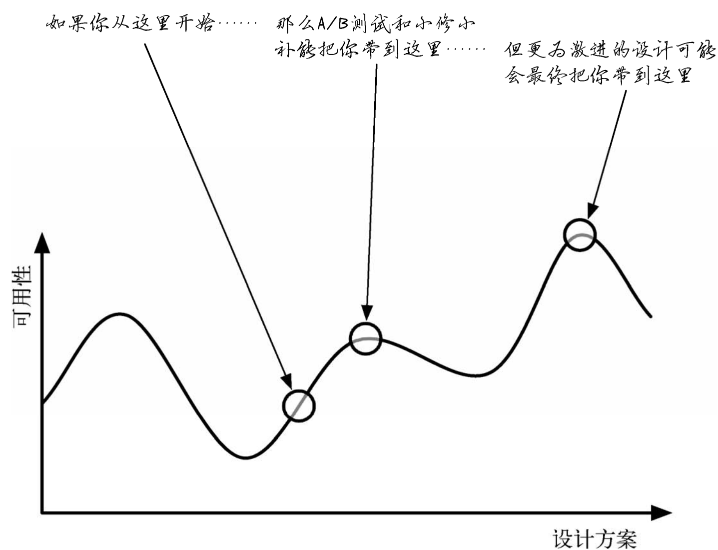
考虑产品所有可行的设计方案。其中某些具有更高的可用性，其他则要差一些。如果你能把这些方案的可用性绘制成曲线图，所得结果应该与下页图类似。

设计方案的改变会引起可用性的改变。

如果设计方案目前的可用性位于上图中间的某个位置，那么小修小补式的迭代变更和 A/B 测试无法让你得到最好的设计方案。试想一下：如果你去爬山，决定一直不停地往上爬，最终肯定会到达山顶，但你绝不会到达附近其他更高的山的山顶。你应该先从现在的山上下来，然后才能到达其他更高的山顶。



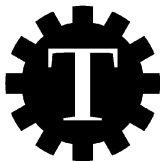
同样，如果你想实现更好的设计方案，那么就要从目前表现较差的设计中抽身出来，开始一个新的设计。



为了避免这种问题，你应该摒弃这种迭代式的设计，尝试一个完全不同的方案。A/B 测试是每个设计师工具箱中非常棒的工具，但一定要知道它的局限性所在。

提示

- A/B 测试能让你对比多个不同的设计方案，找出最有效的那个。
- “有效”的定义取决于用户的目标。
- 收集网站的使用数据很简单，但收集桌面或移动应用程序的使用数据要难一些。记住，要征得用户的同意，并清楚地告诉他们你在做什么。
- 在基于数据制定决策之前，一定要确保结果具有统计意义。
- 在对比高度优化的旧有设计方案和新方案时一定要知道，即便新设计方案在直接对比中稍差一些，但如果你能像对旧方案一样倾注同样多的心血，新方案最终会有更好的表现。



第 34 章

收集产品使用数据

现在，你已经完成了产品的设计，并发布了产品，你的工作是否就完成了呢？当然没有，应用程序和网站的设计永远没有完结的时候。一旦用户开始使用产品，你就要开始估量产品的表现，观察用户用产品来干什么。本章将介绍一些能帮助你完成这些工作的思路和方法。

为什么要收集这些数据？

产品发布之后，用户会以你从未想过的方式使用它。你要知道用户在做什么，这样才知道该如何改进产品。

是否有前提条件？

有。你应该提供可以工作、已发布的产品版本。

34.1　速度

产品的表现很重要。如果产品在使用过程中多次要求用户等待，用户会很讨厌这种经历的。在产品的开发阶段，你可以也应该利用测试数据对产品的表现进行测试。但也要知道，用户会对产品做出一些疯狂的事情——你从未想到过的事情。

如果你正在编写一个 DVD 分类整理程序，让用户可以管理自己拥有的影片。你或许会假设大部分人拥有的 DVD 都不超过 200 张，因此你能保证产品在管理 200 张左右的 DVD 时能工作得很好。但在现实中，产品的部分客户可能是狂热的影片收藏者（这些人正是因为这项爱好才第一时间购买你的产品）。他们用产品管理的 DVD 不是几百张，而是上千张。

或者，假设你正在设计一个面向自由职业者的网站，该网站能让自由职业者查看自己在各种项目和客户上花费的时间。网站在测试过程中的表现很好，因此你就认可了产品在测试中的表现，然后把产品的能力线性地提升，直到你认为能满足发布后客户访问量的要求。但是，这个假设的前提是（你认为）所有用户都是随机地访问网站。试想一下，在你发布了产品之后，用户马上就访问它，会出现什么情况。很多住在同一时区内的用户都在基本相同的时间点开始工作，他们同时打开电脑，访问你的网站，研究某个项目。换句话说，所有人同时点击你的服务，导致了网站访问量达到峰值。用户对网站的访问并不是平均分布在一整天之内，而是会出现访问量高峰和低

谷。产品在访问高峰时的表现如何？会不会因为你的服务器无法处理这种突然出现的访问高峰，导致用户每天第一次访问网站的体验都很糟糕呢？

一旦你完成了产品，用户开始使用它，一定要追踪产品在实际使用中的表现。

至于你的产品应该有多快的速度，请回顾第 23 章。

34.2　退出产品

在观察用户如何使用产品时，你需要关注的一个问题是：用户何时停止使用产品？用户退出产品的时间能告诉你很多关于他们所面对问题的信息。他们是否完成一项任务后就退出了产品？此时，或许产品还没有出现问题。用户是否在执行某一任务的过程中就退出了产品？比如，在你设计的在线商店中购物的过程中，他们是否放弃了已经满了的购物车？如果发生了这种情况，产品的某些方面就存在问题。或许用户开始对你的产品感到厌烦，或许他们不知道该如何完成某项任务，或许他们不知道如果继续下去会发生什么事情。无论如何，产品肯定有问题。

任务执行过程中的退出行为是产品存在可用性问题的指标。为了找到产品潜在的问题，就要追踪这些行为。

34.3　定义“失败”

当产品令用户失望时通常是很明显的，从这些地方很容易收集到一些改进产品的有用数据。

如果用户在访问你的网站时遇到 404 页面，或是类似的错误，你就可以肯定产品出现问题了。记录这些错误和相关产品内容，如果可能，修复这些错误。

如果用户搜索了某些东西，但却没有得到任何结果，或者没有点击任何搜索到的结果，你就要关注一下他们使用的搜索关键词，并找出没有搜索到用户所需内容的原因。你能做些什么来修复这一问题呢？ 无论用户在何时撤销操作，其原因都在于产品没有实现用户想要的东西。记录用户在撤销他们所作的改变之前的行为。

虽然网络冲突并不是严格意义上的可用性问题，但它却对用户体验有着重要的决定性作用。确保用户可以很容易地向你报告冲突问题，只要可能就修复这一问题（有些时候，即便做不到也要修复这一问题）。

游戏设计师可以估量玩家在游戏中的何处死亡（参见第 26 章），与此类似，你也可以估量产品在何处导致用户无法完成任务，并修复相关问题。

34.4　用户行为

在配置完产品之后，你就可以对其进行分析了。用户点击了哪个链接，用户从来没点击过哪

个链接？哪一项功能很受欢迎，哪一项不受欢迎？ClickTale和Crazy Egg之类的服务程序能帮你得到用户行为信息。[①]你可以使用这些数据评估设计方案和产品功能，甚至可以用它来决定要去掉哪些功能。（更多内容，参见第25章。）

以上内容不仅适用于网站产品。你也可以让桌面程序的用户定期向你发送产品使用数据。

只要有人同意向你发送数据，你就要确定哪些数据才是有用的。微软的杰森·哈里斯曾在文章[②]中写道："只要不涉及用户的隐私，我们会收集所有自己认为有趣和有用的信息。"

微软收集的信息包括用户使用的键盘快捷键、用户在不同任务上花费的时间，甚至还收集了用户创建了什么内容，有多少人创建这项内容（比如，用户拥有多少存放邮件的文件夹）等等。你所知道的用户行为越多，就越能形成更好的决策。

如果产品已经发布了一段时间，你也可以邀请有经验的用户执行可用性测试，或者访问一些用户，观察他们真正使用产品的行为（这与你想象用户如何在现实中使用你的产品不同，两者常常有很大的差别）。

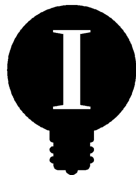
提示

- 一旦用户开始使用你的产品，就要开始估量产品的性能表现。
- 速度是一项很重要的性能，因为你一般很难知道产品在用户手中的实际性能，直到产品为用户所用。
- 记录用户退出产品的行为，因为这表明产品存在一些问题。
- 估量产品的失败表现，如已损坏的链接、空的搜索结果，或是网络冲突之类的现象。
- 观察用户行为，抓住改进产品的最佳时机，需要增加哪些功能，应该去掉哪些功能。
- 用户已经开始使用产品，但这并不是设计的终点。现在正是个大好时机，你需要观察用户实际使用产品的行为，并利用相关信息改进产品。

延伸阅读

杰森·哈里斯是微软Windows用户体验小组的项目管理经理，他有一个人气很高的博客[③]。他经常撰写一些关于微软如何通过产品使用数据制定设计决策的内容。

① 这两个程序分别位于 http://www.clicktale.com 和 http://www.crazyegg.com。
② 关于完整的文章，参见 http://blogs.msdn.com/b/jensenh/archive/2006/04/05/568947.aspx。
③ 参见 http://blogs.msdn.com/b/jensenh。



第 35 章

处理用户反馈

35

产品完成之后，就要开始最难的部分了。用户会以你根本预料不到的方式使用产品。他们会遇到你从未想过的问题，但却把责任归咎于你。我已在第 24 章中介绍了一些处理用户反馈的内容，说明了如何用“五个为什么”来评估用户反馈。在本书的其他章节，我讨论了什么时候该倾听用户的心声，什么时候该斟酌用户的反馈意见。

本章介绍与用户反馈相关的其他内容。

35.1 意料之外的产品使用情景

需要注意的一个问题是，用户可能会以出乎你意料的方式使用产品。在 20 世纪 70 年代到 80 年代间，IBM 极大地低估了个人电脑的市场前景，没有意识到用户会以 IBM 自己都没有想到的方式使用电脑设备。

1989 年，当蒂姆 · 伯纳斯 · 李在欧洲原子核研究委员会（CERN）提出万维网时，他把该技术的目标描述为：“把各种信息连接起来，形成包含有节点的网络，用户可以在其中任意浏览信息。”现在，我们所编写的整个应用程序都在他提出的平台上运行。

让我们看一个距今更近的例子。当播客网站 Odeo 提出要构建一个简单的系统，用以发布人们通常写在 Skype 状态中的各种状态信息时，他们并未预料到用户把 Twitter 当作微博发布站点广泛使用。

问题在于，人们使用产品时并不总是按照你在创建产品时认为或期望用户会使用的方式。这也并不是件坏事，它意味着你可能找到了之前认为不会对产品感兴趣的客户群，也可能发现了以前根本不知道的产品市场。

有三种方法可以处理这一问题。

- 直接忽略。如果附加市场远远小于产品的主要市场，那么你就可以这样做。如果没有按照你预期的方式使用产品的用户对产品非常满意，那太好了！你免费获得了更多的用户。
- 修正产品。如果新市场比产品原有市场大得多，那么改变产品的设计方向专门面向新市场就有一定的价值。
- 折中。如果产品的原有市场和新市场都比较有吸引力，那么就分别为各个市场创建特定的产品版本。

以上三种方法都有其作用所在，但选择哪个方案，一定要三思而后行。

35.2 负面反馈

用户在填写产品的反馈信息时都很刻薄。其原因之一在于，他们认为没有任何人会阅读这些信息。因此，用户把反馈意见表格当成了一个发泄沮丧情绪的渠道。

对此，你不要带有任何个人情绪。毕竟，这些人花费时间对你大吼大叫意味着，他们至少还比较在意你的产品，因此才愿意花时间对你吼。更糟糕的情况是：用户根本就不填写反馈表格。

当我与交互设计师克里斯·克拉克[①]谈到他的设计流程时，他告诉我：[②]

> 我这个人比较奇怪，喜欢负面的反馈信息。出乎你意料之外的用户怨言意味着某些人很在意你的产品，因此才写出了他们的抱怨。当你修复了他们经常反映的问题，他们会变成产品的粉丝。如果能针对某个具体用户修复一个真正的问题，这会带来很大回报。常常令人感到惋惜的是，iOS 开发者并不直接回复应用商店的评论信息，其原因是他们要修复引发问题的根源。
>
> 客户评论确实会对未来产品的发布计划产生些许影响。他们就好像是在哗众取宠。完全忽略你自己的想法，优先考虑用户反馈是愚蠢的……你很快就会发现自己陷入了亨利·福特所谓“我的客户想要一匹更快的马”的境地，但用户反馈确实有其作用。优先考虑呼声最高的功能，然后再考虑其他的。
>
> 有些负面反馈信息根本没有用，但现实就是这样。总有一些刻薄的混蛋无所事事，就喜欢在 iTunes 上给出一颗星的评价。但如果刻薄的词语真的影响到了你，你也没有理由制作一些东西卖给公众。不要把自己当作是音乐家、作家，或者厨师。

不要把负面反馈信息当做是对你的侮辱。这是个机会，你能把怒不可遏的意见用户变成产品的狂热粉丝。

深呼吸一下，不要太在意这些反馈，考虑一下用户实际上想说什么。他会无缘无故这样愤怒吗？你是否能帮助这个用户呢？你是否能做一些事情，让其他人不会再遇到这个用户遇到的问题呢？

我们设计产品，因为想帮助用户实现他们的目标。这也正是你阅读本书的原因所在，目的是让人们生活得更好。负面的用户反馈并不表示你失败了，只能说明你需要付出更多的努力。

提示

- ❑ 用户可能会以你预料不到的方式使用产品。这可能是把产品设计导向新方向的大好时机。
- ❑ 不要因为负面反馈而心灰意冷。你应该利用这些反馈把愤怒的顾客变成产品的粉丝。

① 欲了解该设计师，请访问 http://releasecandidate.com。

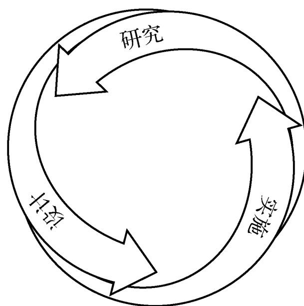
② 访谈详见 http://ignco.de/320。

第 36 章

革命尚未成功

36

这是本书的最后一章，但不是产品开发过程的终点。在本书文前中，我就曾把典型的产品开发描述为下图所示的过程。



本书“实施”部分的最后一章是关于收集数据和处理用户反馈的内容，这并非偶然。这些内容都属于“研究”。也就是说，你又回到了起点。既然你到达了开发流程中“实施”部分的终点，是时候启动一个新的流程，重新考虑所有事情了。用户是否如你想象的那般使用产品？真正的用户是谁？产品是否能解决你想解决的问题？

这一章是本书的终点，但你的工作才刚刚开始！

参 考 文 献

[BB10] Cennydd Bowles and James Box. *Undercover User Experience Design*. New Riders Press, 1 edition, 2010.

[BT10] Nate Bolt and Tony Tulathimutte. *Remote Research*. Rosenfeld Media, 2010.

[Bux07] Bill Buxton. *Sketching User Experiences: Getting the Design Right and the Right Design*. Morgan Kaufmann, 2007.

[CL99] Larry Constantine and Lucy A.D. Lockwood. *Software for Use: A Practical Guide to the Models and Methods of Usage-Centered Design*. Addison-Wesley Professional, 1999.

[Cla10] Josh Clark. *Tapworthy: Designing Great iPhone Apps*. O'Reilly Media, 2010.

[Coo95] Alan Cooper. *About Face: The Essentials of User Interface Design*. John Wiley & Sons, New York, 1995.

[Coo99] Alan Cooper. *The Inmates Are Running the Asylum*. Sams, 1999.

[Cor99] Stanley Coren. *Sensation and Perception*. John Wiley & Sons, 1999.

[Csi02] Mihaly Csikszentmihalyi. *Flow*. Rider, 2002.

[FH10] Jason Fried and David Heinemeier Hansson. *Rework*. Crown Business, 2010.

[FHL09] Jason Fried, David Heinemeier Hansson, and Matthew Linderman. *Getting Real*. 37signals, 2009.

[Fru98] Adrian Frutiger. *Signs and Symbols: Their Design and Meaning*. Watson-Guptill, 1998.

[Gil07] Daniel Gilbert. *Stumbling on Happiness*. Vintage, 2007.

[Hoe06] Robert Hoekman. *Designing the Obvious: A Common Sense Approach to Web Application Design*. New Riders Press, 2006.

[HT00] Andrew Hunt and David Thomas. *The Pragmatic Programmer: From Journeyman to Master*. Addison-Wesley, Reading, MA, 2000.

[Kin00] Stephen King. *On Writing*. Scribner, New York, 2000.

[Kos04] Raph Koster. *A Theory of Fun for Game Design*. Paraglyph Press, 2004.

[Kru09] Steve Krug. *Rocket Surgery Made Easy: The Do-It-Yourself Guide to Finding and Fixing Usability Problems*. New Riders Press, 2009.

[McC94] Scott McCloud. *Understanding Comics*. Harper Paperbacks, 1994.

[MR06] Peter Morville and Louis Rosenfeld. *Information Architecture for the World Wide Web*. O'Reilly Media, 3rd edition, 2006.

[Nor88] Donald A. Norman. *The Design of Everyday Things*. Doubleday/Currency, New York, 1988.

[NY10] Clifford Nass and Corina Yen. *The Man Who Lied to His Laptop: What Machines Teach Us About Human Relationships*. Current Hardcover, London, 2010.

[PA06] John Pruitt and Tamara Adlin. *The Persona Lifecycle*. Morgan Kaufmann, 2006.

[Pal99] Stephen E. Palmer. *Vision Science: Photons to Phenomenology*. The MIT Press, 1999.

[Pap05] Victor Papanek. *Design for the Real World*. Academy Chicago Publishers, 2nd edition, 2005.

[Ras00] Jef Raskin. *The Humane Interface: New Directions for Designing Interactive Systems*. Addison-Wesley Professional, Reading, MA, 2000.

[Ros86] Caroline Rose. *Inside Macintosh*. Addison Wesley, 1986.

[Saf08] Dan Saffer. *Designing Gestural Interfaces*. O'Reilly Media, 2008.

[Sch05] Barry Schwartz. *The Paradox of Choice: Why More Is Less*. Harper Perennial, 2005.

[Sim09] Charlotte Simmonds. *The World's Fastest Flower*. Victoria University Press, 2009.

[SJ94] Michael Steehouder and Carel Jansen. *Quality of Technical Documentation*. Editions Rodopi, 1994.

[Sni03] Carolyn Snider. *Paper Prototyping: The Fast and Easy Way to Define and Refine User Interfaces*. Morgan Kaufmann, 2003.

[Spe09] Donna Spencer. *Card Sorting: Designing Usable Categories*. Rosenfeld Media, 2009.

[Spe10] Donna Spencer. A Practical Guide to Information Architecture. Five Simple Steps, 2010.

[Spo11] Joel Spolsky. *User Interface Design for Programmers*. Apress, 2011.

[SZ03] Katie Salen and Eric Zimmerman. *Rules of Play: Game Design Fundamentals*. The MIT Press, 2003.

[War04] Colin Ware. *Information Visualization: Perception for Design*. Morgan Kaufmann, 2 edition, 2004.


TURING
New Riders
瞬间之美
Web界面设计如何让用户心动
[美] Robert Hoekman, Jr. 著
向怡宁 译
人民邮电出版社
POSTS & TELECOM PRESS


·图灵交互设计丛书·
简约至上
交互式设计四策略
[英] Giles Colborne 著
李松峰 秦绪文 译
人民邮电出版社
POSTS & TELECOM PRESS


TURING
Third Edition
The Non-Designer's Design Book
写给大家看的设计书
（第3版）
[美] Robin Williams 著
苏金国 刘亮 译
有大师指导，人人都能成为设计师！
优秀设计就这么简单：C.R.A.P四条基本原则
生动幽默，图文并茂，让你手不释卷，欲罢不能
人民邮电出版社
POSTS & TELECOM PRESS

“这本书对软件用户体验设计全过程进行了百科全书式的描述，有最新最好的实践案例、内容翔实的设计方法。相见恨晚啊！”

——凯斯•朗，Skitch公司COO和交互设计师

“写可用性话题很难摆脱过分学院派的缺点，但卢卡斯做到了。如果你熟悉基本的可用性概念，又想了解更多的内容，这本书是必读佳作！”

——乔恩•贝尔，Windows Phone的交互设计师

“卢卡斯深刻分析了可用性、用户体验、UI设计等方面许多为人忽视的问题，这是一本必不可少的、权威且极具启发性的著作。”

——保罗•尼夫，Neave交互设计公司交互设计师

用 户 界 面 设 计 百 科 全 书

亚马逊网站读者一致好评

亲爱的界面 让用户乐于使用、爱不释手

Designed for Use

Create Usable Interfaces for Applications and the Web

亲 爱 的 界 面 ， 你 值 得 拥 有 ！

时代在进步，用户对产品的要求也在提高，“能用”已经无法满足用户要求，设计已从一门技术转变为了艺术。可用性则是设计的基石。那么应该如何设计出具有可用性的应用程序和网站，不但让用户乐于使用，而且令其爱不释手呢?

本书作者根据多年交互设计经验，把用户界面设计分为研究、设计与实施三个重要阶段，以独具匠心的设计视角，介绍了如何将可用性融入设计的各个阶段，使界面让用户一见倾心。

书中内容分为两类：设计技术和设计创意。设计技术章节主要介绍适用于不同设计阶段、行之有效的用户界面设计技术；在设计创意章节里，作者结合自己多年的设计经验和最新的设计理论，向读者提供了激发设计灵感的各种方法，并给出了相关设计建议。

海量的设计与评估技巧、丰富的插图与设计案例、循序渐进的设计指导，加之对心理学研究成果的运用，让本书如百科全书般详尽实用。

本书适用于设计人员、开发人员、产品经理、专业软件开发工程师和用户界面设计师。

The Pragmatic Programmers

图灵社区：www.ituring.com.cn

新浪微博：@图灵教育 @图灵社区

反馈/投稿/推荐邮箱：contact@turingbook.com

热线：(010)51095186转604

分类建议 计算机/UI设计

人民邮电出版社网址：www.ptpress.com.cn



ISBN 978-7-115-29639-9

定价：49.00元